

illumina®

Illumina DRAGEN Bio-IT Platform v4.2

User Guide

ILLUMINA PROPRIETARY
Document #200033181v02
April 2024

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

This document and its contents are proprietary to Illumina, Inc. and its affiliates ("Illumina"), and are intended solely for the contractual use of its customer in connection with the use of the product(s) described herein and for no other purpose. This document and its contents shall not be used or distributed for any other purpose and/or otherwise communicated, disclosed, or reproduced in any way whatsoever without the prior written consent of Illumina. Illumina does not convey any license under its patent, trademark, copyright, or common-law rights nor similar rights of any third parties by this document.

The instructions in this document must be strictly and explicitly followed by qualified and properly trained personnel in order to ensure the proper and safe use of the product(s) described herein. All of the contents of this document must be fully read and understood prior to using such product(s).

FAILURE TO COMPLETELY READ AND EXPLICITLY FOLLOW ALL OF THE INSTRUCTIONS CONTAINED HEREIN MAY RESULT IN DAMAGE TO THE PRODUCT(S), INJURY TO PERSONS, INCLUDING TO USERS OR OTHERS, AND DAMAGE TO OTHER PROPERTY, AND WILL VOID ANY WARRANTY APPLICABLE TO THE PRODUCT(S).

ILLUMINA DOES NOT ASSUME ANY LIABILITY ARISING OUT OF THE IMPROPER USE OF THE PRODUCT(S) DESCRIBED HEREIN (INCLUDING PARTS THEREOF OR SOFTWARE).

© 2024 Illumina, Inc. All rights reserved.

All trademarks are the property of Illumina, Inc. or their respective owners. For specific trademark information, refer to www.illumina.com/company/legal.html.

Table of Contents

Getting Started	1
DRAGEN Bio-It Platform Overview	1
Run Requirements	3
Software Installation	5
System Updates	6
License Usage	6
Running the System Check	7
Running Your Own Test	8
Generate a Reference	8
Prepare a Reference Genome	11
Prepare a Custom Multigenome Reference	26
Determine Input and Output File Locations	29
Process Your Input Data	29
Example Commands for Processing FASTQ Data	29
DRAGEN Host Software	54
Command-line Options	54
Operating Modes	57
Output Options	58
Input Options	61
Autogenerated MD5SUM for BAM and CRAM Output Files	74
Configuration Files	74
Cloud Authentication and Licensing	74
DRAGEN Reference Support	76
DRAGEN DNA Pipeline	79
DNA Mapping	79
DNA Aligning	82
DRAGEN Graph Mapper	92
Read Trimming	92
DRAGEN FastQC	101
Mapping with ALT-Aware	105
Sorting	107
Filter Duplicate Variants	107
Duplicate Marking	109
Small Variant Calling	110
Copy Number Variant Calling	177

CNV with SV Support	209
Multisample CNV Calling	212
Somatic CNV Calling	215
Repeat Expansion Detection with ExpansionHunter	228
Paralog Calling	232
Structural Variant Calling	290
Structural Variant De Novo Quality Scoring	317
Ploidy Estimator	319
Ploidy Caller	321
QC Metrics and Coverage/Callability Reports	325
DRAGEN HLA Caller	349
Biomarkers	355
Downsampling	373
Virtual Long Read Detection	374
Unique Molecular Identifiers	377
Multicaller Workflows	391
Indel Re-alignment (Beta)	396
Star Allele Caller	398
High Coverage Analysis	404
CheckFingerprint Software Design	405
Population Haplotyping (Beta)	409
Explify Pipeline	425
Explify Analysis Pipeline	425
Kmer Classifier	446
Recipes	453
Germline WES	453
Germline WGS	458
Somatic UMI Tumor Normal	462
Somatic UMI Tumor Only	476
Somatic WES Tumor Normal	488
Somatic WES Tumor Only	499
Somatic WGS Tumor Normal	510
Somatic WGS Tumor Only	520
cfDNA UMI with Enrichment on FFPE Samples	531
DRAGEN RNA Pipeline	537
Input Files	537
RNA Alignment	540

Alignment Output	540
RNA Alignment Options	544
Duplicate Marking	546
Downsampling	546
Ribosomal RNA Filtering	546
PolyA Trimming	547
MAPQ Scoring	547
Gene Fusion Detection	548
Gene Expression Quantification	560
RNA Variant Calling	564
DRAGEN Single-Cell Pipeline	566
Multiomics Pipeline	566
ATAC Pipeline	572
RNA Pipeline	583
DRAGEN Methylation Pipeline	600
Mapping Method Options	601
Build a Methylation Hash Table	602
DRAGEN Methylation Calling	603
Using Bismark for Methylation Calling	605
Sort and Duplicate Reads Options	605
Unique Molecular Identifiers (UMI) Support	606
Using TAPS Support	607
Methylation-Related BAM Tags	607
Methylation Cytosine and M-Bias Reports	608
Output Metrics	609
DRAGEN Amplicon Pipeline	609
Amplicon BED File	610
DRAGEN DNA Amplicon Settings	611
DRAGEN RNA Amplicon Settings	612
Tools and Utilities	613
BCL Data Conversion	613
Monitoring System Health	650
Illumina Annotation Engine	653
DRAGEN ORA Compression and Decompression	668
Hardware-Accelerated Compression and Decompression	673
Usage Reporting	673

Troubleshooting	675
How to Determine if the System is Hanging	675
Sending Diagnostic Information to Illumina Support	675
Resetting Your System after a Crash or Hang	675
Command-Line Options	677
Resources & References	733
Revision History	733

Getting Started

Before you begin, make sure that the Illumina server is turned on and that you are logged in.

This Getting Started section helps you to start processing data as quickly as possible.

DRAGEN provides tests you can run to make sure that your DRAGEN system is properly installed and configured. Before running the tests, make sure that the DRAGEN server has adequate power and cooling, and is connected to a network that is fast enough to move your data to and from the machine with adequate performance.

DRAGEN Bio-IT Platform Overview

The Illumina DRAGEN™Bio-IT Platform is based on the highly reconfigurable DRAGEN Bio-IT Processor, which is integrated on a Field Programmable Gate Array (FPGA) card and is available in a preconfigured server that can be seamlessly integrated into bioinformatics workflows. The platform can be loaded with highly optimized algorithms for many different NGS secondary analysis pipelines, including the following:

- Whole genome
- Exome
- RNA-Seq
- Methylome
- Cancer

All interaction is accomplished via DRAGEN software that runs on the host server and manages all communication with the DRAGEN board.

This user guide summarizes the technical aspects of the system and provides detailed information for all DRAGEN command line options.

If you are working with DRAGEN for the first time, Illumina recommends that you first read the [Getting Started on page 1](#) section, which provides a short introduction to DRAGEN, including running a test of the server, generating a reference genome, and running example commands.

Input Requirements

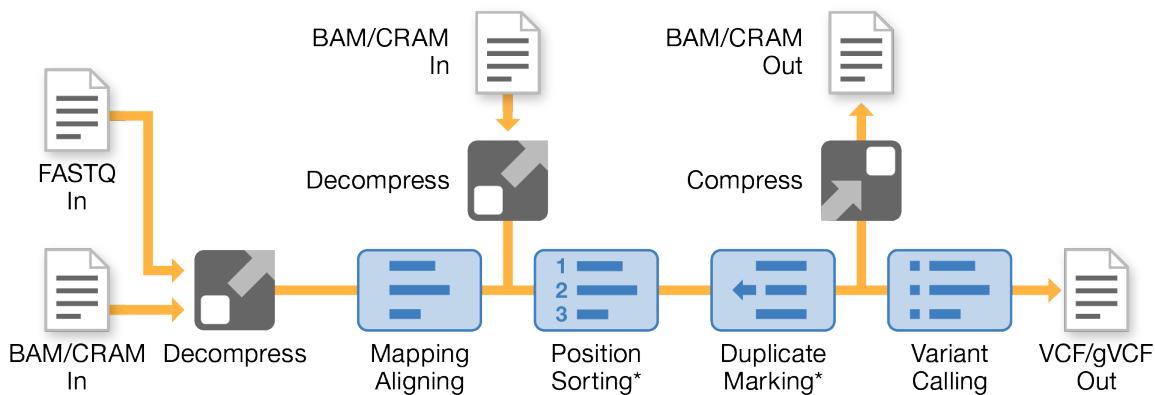
The following are the supported input specifications for DRAGEN.

Specifications	Requirement
Supported Input Files	cBCL, FASTQ, BAM, CRAM, GVCF

Specifications	Requirement
Upper Limit	300x coverage of human Whole Genome sequencing data. 200x/100x coverage of human T/N.

DRAGEN DNA Pipeline

Figure 1 DRAGEN DNA Pipeline



* Optional

The DRAGEN DNA Pipeline accelerates the secondary analysis of NGS data. For example, the time taken to process an entire human genome at 30x coverage is reduced from approximately 10 hours (using the current industry standard, BWA-MEM+GATK-HC software) to approximately 20 minutes. Time scales linearly with coverage depth.

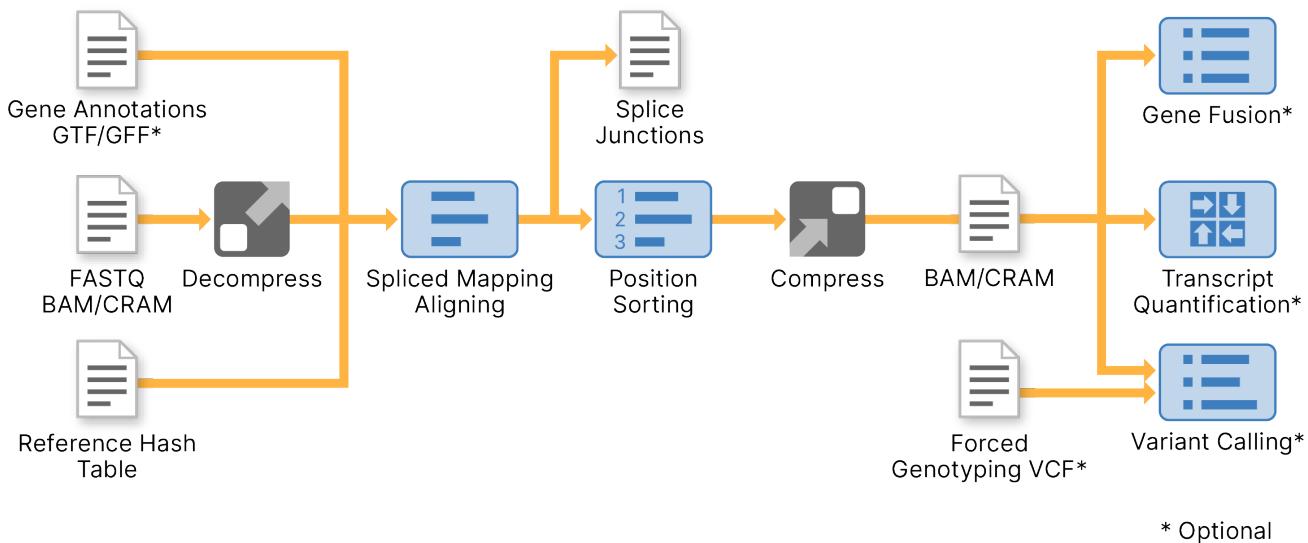
These pipelines harness the tremendous power of the DRAGEN Bio-It Platform and include highly optimized algorithms for mapping, aligning, sorting, duplicate marking, and haplotype variant calling. They also use platform features such as hardware-accelerated compression and optimized BCL conversion, together with the full set of platform tools.

Unlike all other secondary analysis methods, DRAGEN DNA Applications do not reduce accuracy to achieve speed improvements. Accuracy for both SNPs and INDELs is improved over that of BWA-MEM+GATK-HC in side-by-side comparisons.

In addition to haplotype variant calling, the pipeline supports calling of copy number and structural variants as well as detection of repeat expansions.

DRAGEN RNA Pipeline

DRAGEN includes an RNA (splicing-aware) aligner, as well as RNA-specific analysis components for gene expression quantification and gene fusion detection.



The DRAGEN RNA Pipeline shares many components with the DNA Pipeline. Mapping of short seed sequences from RNA reads is performed similarly to mapping DNA reads. In addition, splice junctions (the joining of noncontiguous exons in RNA transcripts) near the mapped seeds are detected and incorporated into the full read alignments.

DRAGEN Methylation Pipeline

The DRAGEN Methylation Pipeline provides support for automating the processing of bisulfite sequencing data to generate a BAM with the tags required for methylation analysis and reports detailing the locations with methylated cytosines.

Run Requirements

The Analysis CUG support pages on the Illumina [support site](#) provide additional information.

DRAGEN Bio-It Platform Analysis app app requires the following files to be present in the run folder to complete analysis:

- BCL files (*.bcl, *.cbcl)
- Filter files (*.filter)
- Position files (*.locs, *.clocs, *.s.locs)
- Aggregated files (*.bci)
- Run info file (*.xml)

- Config.xml—The config.xml file is only required for data produced by some systems. See the DRAGEN pages on the Illumina support site for more information.

Sequencing Data

DRAGEN requires the following files to run. Different inputs might be required based on the sequencing system used to produce the data.

Input	Description
Base call files (*.bcl.bgzf)	Base call (BCL) files are compressed with the gzip (*.gz) or blocked GNU zip (*.bgzf) format.
Base call index files (*.bci)	Base call index files (BCI) files contain one record per lane in binary format. BCI files are acceptable as input by DRAGEN but are not used for analysis.
Concatenated base call files (*.cbcl)	Concatenated base call (CBCL) files contain aggregated BCL data. Tiles from the same lane and surface are aggregated into one CBCL file for each lane and surface.
Filter files (*.filter)	Filter files are binary files that specify whether a given cluster passed filter.
Location files (*.locs, s.locs)	Location files (LOCS) are binary files that contain the cluster positions on the flow cell. CLOCS files are compressed versions of LOCS files.
Run information file (RunInfo.xml)	The run information file resides at the root level of the output folder. The file contains the run name, number of cycles, whether a read is an Index Read, and the number of lanes, swaths, and tiles. If this file does not exist in the output folder, the software produces an error.
Configuration file (*.xml)	The configuration file resides in the BaseCalls folder and contains metadata on the sequencing run. The file is in XML format.

Running DRAGEN

Use the following information when running DRAGEN for BCL conversion.

To prevent a disconnection or terminal closure when executing BCL conversion through the command line, enter `nohup` before the executable you want to run.

Ulimit Settings

DRAGEN requires high ulimit settings for both the number of open files allowed and maximum user process. If a run fails due to maximum user processes being set too low, an error message stating that the resource is temporarily unavailable occurs. By default, DRAGEN attempts to set the ulimit soft limit for the number of open files (`ulimit -n`) to 65535 and the maximum user processes to 32768. If those

values exceed the hard limits of the system, the soft limit is set to the hard limit.

If more than 10,000 samples are provided, then `ulimit -n` is set to 720000.

Missing File Handling

If `--strict-mode` is set to false, DRAGEN executes certain behaviors when it finds missing or corrupt files. The following are the possible behaviors according to file type and status.

File Type	Status	Behavior
*.bcl	Missing or corrupt	All base calls of the cycle in the corresponding lane and tile are replaced with N and a quality score of #.
*.cbcl	Missing or corrupt	All base calls of the cycle in the corresponding lane and surface are replaced with N and a quality score of #.
*.cbcl	Corrupt	All base calls of the cycle in the corresponding lane and tiles are replaced with N and a quality score of #.
*.locs	Missing or corrupt	Produce FASTQ files with an automatically generated unique header for all reads in the corresponding lane and tiles.
*.filter	Missing or corrupt	No FASTQ entries produced for any reads in the corresponding lane and tiles.
*.bci lane	Missing or corrupt	No FASTQ entries produced for any reads in the corresponding lane and tiles.

Software Installation

If you are already running the latest version of the DRAGEN software and hardware, you can skip ahead to [Running the System Check on page 7](#).

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"
```

System Updates

You can update any software on the DRAGEN server. You can download new versions of DRAGEN software from the DRAGEN Bio-It Platform support site.

Kernel packages should come from CentOS Updates only. Using experimental kernels or compiling the kernel from source is not recommended.

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
(100000000000 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.

Running the System Check

After turning on the server, you can make sure that your DRAGEN server is functioning properly by running `/opt/edico/self_test/self_test.sh`, which does the following:

- Automatically indexes chromosome M from the hg19 reference genome
- Loads the reference genome and index
- Maps and aligns a set of reads
- Saves the aligned reads in a BAM file
- Asserts that the alignments exactly match the expected results

Each server ships with the test input FASTQ data for this script, which is located in `/opt/edico/self_test`. The system check takes approximately 25–30 minutes.

The following example shows how to run the script and shows the output from a successful test.

```
[root@edico2 ~]# /opt/edico/self_test/self_test.sh
-----
test hash creating
test hash created
-----
reference loading /opt/edico/self_test/ref_data/chrM/hg19_chrM
reference loaded
-----
real0m0.640s
user0m0.047s
sys0m0.604s
not properly paired and unmapped input records percentages: PASS
-----
md5sum check dbam sorted: PASS
-----
SELF TEST COMPLETED
SELF TEST RESULT : PASS
```

If the output BAM file does not match expected results, then the last line of the above text is as follows:

```
SELF TEST RESULT : FAIL
```

If you experience a FAIL result after running this test script immediately after turning on your DRAGEN server, contact Illumina Technical Support.

Running Your Own Test

When you are satisfied that your DRAGEN system is performing as expected, you are ready to run some of your own data through the machine, as follows:

- Load the reference table for the reference genome
- Determine location of input and output files
- Process input data

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 the Reference Genome

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
```

This command loads the binary reference genome into memory on the DRAGEN board, where it is used for processing any number of input data sets. You do not need to reload the reference genome unless you restart the system or need to switch to a different reference genome. It can take up to a minute to load a reference genome.

DRAGEN checks whether the specified reference genome is already resident on the board. If it is, then the upload of the reference genome is automatically skipped. You can force reloading of the same reference genome using the *force-load-reference (-l)* command line option.

The command to load the reference genome prints the software and hardware versions to standard output. For example:

```
DRAGEN Host Software Version 01.001.035.01.00.30.6682 and
Bio-IT Processor Version 0x1001036
```

After the reference genome has been loaded, the following message is printed to standard output:

```
DRAGEN finished normally
```

Prepare a Reference Genome

Before a reference genome can be used with DRAGEN, the reference genome must be converted from FASTA format into a custom binary format. The options used in this preprocessing step offer tradeoffs between performance and mapping quality.

Pre-built DRAGEN reference genomes are available to download in the Illumina customer portal. If the performance and mapping quality are adequate, there is a good chance that you can work with the supplied reference genomes. Depending on your read lengths and other particular aspects of your application, you may be able to improve mapping quality and/or performance by tuning the reference preprocessing options.

Hash Table Background

The DRAGEN mapper extracts overlapping seeds (subsequences or K-mers) from each read. DRAGEN looks up the seeds in a hash table that resides in memory on the PCIe card to identify locations in the reference genome where the seeds match. Hash tables are ideal for fast lookups of exact matches. The DRAGEN hash table must be constructed from a chosen reference genome using the `dragen --build-hash-table` option, which extracts many overlapping seeds from the reference genome, populates them into records in the hash table, and saves the hash table as a binary file.

Automatic Reference Detection

DRAGEN will attempt to detect the provided reference to automatically apply recommended resources and settings. There are four human reference that DRAGEN can detect: hg38, hg19, hs37d5, and chm13v2. DRAGEN will compare the names and lengths of the contigs from the provided reference to the names and lengths of the autosomes(1-22) and sex chromosomes from these four human references. Auto-detection will succeed if any of the contigs from the provided reference match any of the autosomes or sex chromosomes from exactly one of hg38, hg19, hs37d5, or chm13v2. If contigs from the provided reference matched against more than one references, auto-detection will fail and no resources will be applied automatically.

Naming Conventions

In order for DRAGEN to correctly detect the provided reference, it is important to use the standard naming conventions for each of the four human assemblies that DRAGEN detects:

Assembly	Autosome and Sex Chromosome Names
hg38, hg19, chm13v2	chr1-chr22, chrX, chrY
hs37d5	1-22, X, Y

Graph References

See [DRAGEN Graph Mapper on page 92](#) for information on the graph mapping method.

Graph references can be build automatically for hg38, hg19, hs37d5, and chm13 by setting `--ht-apply-graph=true`. The hash table builder will automatically apply graph resources for any detected autosomes and sex chromosomes that have the same base content as in the standard assembly. Customizations to the contigs will inhibit automatic graph resource application.

Reference Seed Interval

The size of the DRAGEN hash table is proportionate to the number of seeds populated from the reference genome. The default is to populate a seed starting at every position in the reference genome, which is roughly 3 billion seeds from a human genome. This default requires at least 32 GB of memory on the DRAGEN PCIe board.

To operate on larger, nonhuman genomes or to reduce hash table congestion, you can populate less reference seeds by using the `--ht-ref-seed-interval` option to specify an average reference interval. The default interval for 100% population is `--ht-ref-seed-interval 1`. You can specify 50% population with `--ht-ref-seed-interval 2`. The population interval does not need to be an integer. For example, `--ht-ref-seed-interval 1.2` indicates 83.3% population with mostly 1-base and some 2-base intervals to achieve a 1.2 base interval on average.

Hash Table Occupancy

Hash tables are allocated a certain size, but always retain some empty records, so they are less than 100% occupied. Empty space is important for quick access to the DRAGEN hash table. Records are pseudorandomly placed in the hash table, which results in an abnormally high number of records in some places. DRAGEN recommends approximately 90% occupancy as an upper bound. As the percentage of empty space approaches zero, the congested regions can become large and queries by the DRAGEN mapper for some seeds can become increasingly slow.

Hash Table / Seed Length

The hash table is populated with reference seeds of a single common length. This primary seed length is controlled with the `--ht-seed-len` option, which defaults to 21.

The longest primary seed supported is 27 bases when the table is 8–31.5 GB in size. Generally, longer seeds are better for run time performance, and shorter seeds are better for mapping quality (success rate and accuracy). A longer seed is more likely to be unique in the reference genome, which facilitates fast mapping without needing to check many alternative locations. However, a longer seed is more likely to overlap with a deviation, such as a variant or sequencing error, from the reference. This prevents successful mapping by an exact match of that seed; however, another seed from the read could still map, and there are fewer long seed positions available in each read.

Longer seeds are more appropriate for longer reads, because there are more seed positions available to avoid deviations.

Table 1 Seed Length Recommendations

Value for <i>-ht-seed-len</i>	Read Length
21	100 bp to 150 bp
17 to 19	shorter reads (36 bp)
27	250+ bp

Hash Table / Seed Extensions

Due to repetitive sequences, some seeds of any given length can match more than one location in the reference genome. DRAGEN uses a unique mechanism called seed extension to successfully map such high-frequency seeds. When DRAGEN determines that a primary seed occurs at more than one reference location, DRAGEN extends the seed by a number of bases at both ends until the length is greater and more unique in the reference.

For example, a 21-base primary seed could be extended by seven bases at each end to a 35-base extended seed. A 21-base primary seed could match 100 places in the reference. 35-base extensions of these 100 seed positions could divide into 40 groups of 1–3 identical 35-base seeds. DRAGEN supports iterative seed extensions, which are automatically generated when a large set of identical primary seeds contains various subsets that are best resolved by different extension lengths.

By default the maximum extended seed length is equal to the primary seed length plus 128. To modify the maximum extended seed length, use the `--ht-max-ext-seed-len` option. For example, for short reads, DRAGEN recommends setting the maximum extended seed shorter than the read length, because extensions longer than the whole read can never match.

You can also tune how aggressively seeds are extended using the following options. These options are for advanced usage only.

- `--ht-cost-coeff-seed-len`
- `--ht-cost-coeff-seed-freq`
- `--ht-cost-penalty`
- `--ht-cost-penalty-incr`

There is a tradeoff between extension length and hit frequency. You can achieve faster mapping by using longer seed extensions to reduce seed hit frequencies. You can achieve more accurate mapping by avoiding seed extensions or keeping extensions short, while tolerating the resulting higher hit frequencies. Shorter extensions can benefit mapping quality both by fitting seeds better between SNPs and by finding more candidate-mapping locations at which to score alignments. The default extension settings, along with default seed frequency settings, lean aggressively toward mapping accuracy with relatively short seed extensions and high hit frequencies.

The defaults for the seed frequency options are as follows.

Option	Default
--ht-cost-coeff-seed-len	1
--ht-cost-coeff-seed-freq	0.5
--ht-cost-penalty	0
--ht-cost-penalty-incr	0.7
--ht-max-seed-freq	16
--ht-target-seed-freq	4

Seed Frequency Limit and Target

One primary or extended seed can match multiple places in the reference genome. All such matches are populated into the hash table, and then retrieved when the DRAGEN mapper looks up a corresponding seed extracted from a read. The multiple reference positions are then considered and compared to generate aligned mapper output. However, DRAGEN enforces a limit on the number of matches, or frequency, of each seed. You can modify the frequency using the `--ht-max-seed-freq` option. By default, the frequency limit is 16. When DRAGEN encounters a seed with higher frequency, DRAGEN extends the seed to a sufficiently long secondary seed that the frequency of any particular extended seed pattern falls within the limit. If a maximum seed extension still exceeds the limit, the seed is rejected and is not populated into the hash table. Instead, DRAGEN populates a single High Frequency record.

This seed frequency limit does not tend to impact DRAGEN mapping quality notably due to the following.

- Seeds are rejected only when seed extension fails. Only extremely high-frequency primary seeds, typically with many thousands of matches are rejected. The seeds are not useful for mapping.
- There are other seed positions to check for in a given read. If another seed position is unique enough to return one or more matches, the read can still be properly mapped. However, if all seed positions are rejected as high frequency, this could mean that the entire read matches many reference positions. If the read were mapped it would be an arbitrary mapping, with a very low or zero MAPQ.

You can increase or decrease the frequency up to a maximum of 256. A higher frequency limit tends to marginally increase the number of reads mapped, especially for short reads, but the additional mapped reads could have very low or zero MAPQ. This also tends to slow down DRAGEN mapping, because correspondingly large numbers of possible mappings are occasionally considered.

In addition to a frequency limit, you can specify a target seed frequency using the `--ht-target-seed-freq` option. This target frequency is used when extensions are generated for high frequency primary

seeds. Extension lengths are chosen with a preference toward extended seed frequencies near the target. The default value is 4, which means that DRAGEN is biased toward generating shorter seed extensions than necessary to map seeds uniquely.

Handling Decoy Contigs

The DRAGEN hash table builder automatically detects the absence of the decoy contigs from the reference and adds the decoy contigs to the FASTA file, prior to building the hash table. The decoys file is found at `/opt/edico/liftover/hs_decoys.fa`. If the reference is missing the decoy contigs, then the reads that map to the decoy contigs are artificially marked as unmapped in the output BAM because the original reference does not have the decoy contig. This results in an artificially lower mapping rate. However, the removal of false positive caused by decoy reads results in improved accuracy of variant calling.

Illumina recommends using this feature by default. However, you can set the `--ht-suppress-decoys` option to true to suppress adding these decoys to the hash table.

The table below describes the difference in behavior between older DRAGEN versions (2.6 and earlier) and DRAGEN 3.x versions with respect to the handling of decoy contigs in the hash table builder:

DRAGEN Behavior	DRAGEN 2.6 and earlier versions	DRAGEN 3.x
Reference does not include the decoy contigs (eg, hg19)	<p>Decoy reads mismatch elsewhere in the genome due to the lack of contigs in the reference.</p> <ul style="list-style-type: none"> Artificially higher mapping rate. False positive calls in noisy regions to which the decoy contigs are mismapped. 	<p>DRAGEN automatically detects the absence of the decoy contig from the reference and adds it to the FASTA file.</p> <ul style="list-style-type: none"> Artificially lower mapping rate (because decoy reads which map to the decoy contigs are artificially marked as unmapped in the output BAM (because the original reference does not have the decoy contig)). False positive calls are avoided thanks to adding the decoy contigs under the hood. Therefore this helps variant calling.
Reference includes the decoy contigs (eg, hs37d5)	<p>Decoy reads map to the decoy contigs.</p> <ul style="list-style-type: none"> High mapping rate No false positive calls caused by decoy reads because decoy reads map to the right place 	<p>Decoy reads map to the decoy contig.</p> <ul style="list-style-type: none"> High mapping rate No false positive calls caused by decoy reads because decoy reads map to the right place

References with ALT Contigs

When building a reference hash table from a FASTA with ALT contigs, it may be desired to mask certain regions of high similarity, or to establish a liftover relationships between primary and alternate contigs. The recommended approach is masking, as described in the Map-Align section. When hg19 or hg38 alt contigs are detected, the hash table builder will require a liftover file or a bed file to mask the alt contigs. If none are provided, a mask bed file from `/opt/edico/fasta_mask/` will be used automatically.

Masked References

DRAGEN has adopted a masked approach to handle native reference ALT contigs, where strategic regions are masked to increased accuracy. The hash table builder will build the mapper hash table as if the regions that were specified in the argument for `ht-mask-bed` were masked with N's. The hash table builder will only allow setting one of `ht-mask-bed` or `ht-alt-liftover`. Each line in the bed file is expected to contain a contig name, start position (0-based), and end position (1-based), separated by a single tab or space. Lines that start with # are ignored by the hash table builder to allow commenting. Any line with a contig name that is not found in the input FASTA is skipped and logged to the DRAGEN log file. Likewise, lines that describe empty intervals are skipped. If all lines are skipped this way, the hash table builder will issue an error and abort, unless the mask bed file was automatically applied (see Automatic masking). The hash table builder will always issue an error and abort if an interval described in the BED file is outside of the range of the corresponding contig in the fasta. Lines that are not skipped are written to a file called `mask.bed` that will be present in the hash table output directory, and whose digest will appear in `hash_table.cfg`. This file is used when a reference is loaded to the FPGA card to dynamically mask `reference.bin`.

Automatic masking

When running from a fasta for which hg38 or hg19 is detected (See Automatic Reference Detection), and no argument for `ht-mask-bed` or `ht-alt-liftover` was provided, the hash table builder will automatically apply the corresponding bed file for the detected reference from `/opt/edico/fasta_mask/`.

The hash table builder will identify alt contigs by name. When running from an input fasta that contains alt contig with standard names but modified base content, it is recommended to suppress automatic masking by setting `ht-suppress-mask=true` or by passing a custom mask bed file to `ht-mask-bed`.

Liftover Based ALT-Aware Hash Tables

While masking is the recommended approach to dealing with ALT contigs, DRAGEN also supports a liftover based method. To enable liftover based ALT-aware mapping in DRAGEN, build the hash table with a liftover file by using the `--ht-alt-liftover` option. The hash table builder classifies each reference sequence as primary or alternate based on the liftover file, and packs primaries before alternates in `reference.bin`. SAM liftover files for hg38DH and hg19 are in the `/opt/edico/liftover` folder.

Custom Liftover Files

Custom liftover files can be used in place of those provided with DRAGEN. Liftover files must be SAM format, but no SAM header is required. SEQ and QUAL fields can be omitted (*). Each alignment record should have an alternate haplotype reference sequence name as QNAME, indicating the RNAME and POS of its liftover alignment in a destination (normally primary assembly) reference sequence.

Reverse-complemented alignments are indicated by bit 0x10 in FLAG. Records flagged unmapped (0x4) or secondary (0x100) are ignored. The CIGAR may include hard or soft clipping, leaving parts of the ALT contig unaligned.

A single reference sequence cannot serve as both an ALT contig (appearing in QNAME) and a liftover destination (appearing in RNAME). Multiple ALT contigs can align to the same primary assembly location. Multiple alignments can also be provided for a single ALT contig (extras optionally be flagged 0x800 supplementary), such as to align one portion forward and another portion reverse-complemented. However, each base of the ALT contig only receives one liftover image, according to the first alignment record with an M CIGAR operation covering that base.

SAM records with QNAME missing from the reference genome are ignored, so that the same liftover file may be used for various reference subsets, but an error occurs if any alignment has its QNAME present but its RNAME absent.

Command Line Options

Use the `--build-hash-table` option to transform a reference FASTA into the hash table for DRAGEN mapping. The option takes as input a FASTA file with multiple reference sequences being concatenated and a preexisting output directory. DRAGEN generates the following set of files:

<code>reference.bin</code>	The reference sequences, encoded in 4 bits per base. Four-bit codes are used, so the size in bytes is roughly half the reference genome size. In between reference sequences, N are trimmed and padding is automatically inserted. For example, hg19 has 3,137,161,264 bases in 93 sequences. This is encoded in 1,526,285,312 bytes = 1.46 GB, where 1 GB means 1 GiB or 2^{30} bytes.
<code>hash_table.cmp</code>	Compressed hash table. The hash table is decompressed and used by the DRAGEN mapper to look up primary seeds with length specified by the <code>--ht-seed-len</code> option and extended seeds of various lengths.
<code>hash_table.cfg</code>	A list of parameters and attributes for the generated hash table in a text format. This file provides key information about the reference genome and hash table.
<code>hash_table.cfg.bin</code>	A binary version of <code>hash_table.cfg</code> used to configure DRAGEN.

hash_table_stats.txt	A text file listing extensive internal statistics on the constructed hash including the hash table occupancy percentages. This table is for information purposes and is not used by other tools.
mask.bed	Present only for masked hash tables. A tab delimited bed file that describes the masked regions. Contains all lines from the input bed file that are not comment lines, lines that describe empty intervals, or lines with contig names that were not found in the input FASTA.

Build command usage is as follows.

```
dragen --build-hash-table true [options] --ht-reference <reference.fasta> --output-directory <outdir>
```

The sections that follow provide information on the options for building a hash table.

Input/Output Options

The `--ht-reference` and `--output-directory` options are required for building a hash table. The `--ht-reference` option specifies the path to the reference FASTA file, while `--output-directory` specifies a preexisting directory where the hash table output files are written. DRAGEN recommends organizing various hash table builds into different folders. As a best practice, folder names should include any nondefault parameter settings used to generate the contained hash table. The sequence names in the reference FASTA file must be unique.

Primary Seed Length

The `--ht-seed-len` option specifies the initial length in nucleotides of seeds from the reference genome to populate into the hash table. At run time, the mapper extracts seeds of the same length from each read and looks for exact matches unless seed editing is enabled in the hash table.

The maximum primary seed length is a function of hash table size. The limit is $k=27$ for table sizes from 16–64 GB, which covers typical sizes for whole human genome or $k=26$ for sizes from 4–16 GB.

The minimum primary seed length depends mainly on the reference genome size and complexity. The seed length needs to be long enough to resolve most reference positions uniquely. For whole human genome references, hash table construction typically fails with $k < 16$. The lower bound could be smaller for shorter genomes or higher for less complex (more repetitive) genomes. The uniqueness threshold of `--ht-seed-len 16` for the 3.1 Gbp human genome can be understood intuitively because $\log_4(3.1\text{G}) \approx 16$, so it requires at least 16 choices from four nucleotides to distinguish 3.1 G reference positions.

Accuracy Considerations

For read mapping to succeed, at least one primary seed must match exactly or with a single SNP when edited seeds are used. Shorter seeds are more likely to map successfully to the reference, because more seeds can fit in each read and seeds are less likely to overlap variants or have sequencing errors.

However, very short seeds can sometimes reduce mapping accuracy. Very short seeds often map to multiple reference positions and can lead the mapper to consider more false mapping locations. Due to imperfect modeling of mutations and errors by Smith-Waterman alignment scoring and other heuristics, occasionally these noise matches are reported.

Run time quality filters such as `--Aligner.aln_min_score` can control the accuracy issues with very short seeds.

Speed Considerations

Shorter seeds tend to slow down mapping because the seeds map to more reference locations, which result in additional work, such as using Smith-Waterman alignments to determine the best result. This effect is most pronounced when primary seed length approaches the uniqueness threshold of the reference genome, eg, K=16 for whole human genome.

Application Considerations

- **Read Length**—Generally, shorter seeds are appropriate for shorter reads, and longer seeds for longer reads. Within a short read, a few mismatch positions from variants or sequencing errors can chop the read into only short segments that match the reference, so that only a short seed can fit between the differences and match the reference exactly. For example, in a 36 bp read, one SNP in the middle can block seeds longer than 18 bp from matching the reference. In a 250 bp read, 15 SNPs are required to exceed a 0.01% chance of blocking 27 bp seeds.
- **Paired Ends**—The use of paired end reads can improve mapping accuracy for longer reads. DRAGEN uses paired end information to improve mapping accuracy, including with rescue scans that search the expected reference window when only one mate has seeds mapping to a given reference region. Thus, paired end reads have twice the opportunity for an exact matching seed to find their correct alignments.
- **Variant or Error Rate**—When read differences from the reference are more frequent, shorter seeds could be required to fit between the difference positions in a given read and match the reference exactly.
- **Mapping Percentage Requirement**—If the application requires a high percentage of reads to be mapped (even at low MAPQ), short seeds may be helpful. Some reads that do not match the reference well are more likely to map using short seeds to find partial matches to the reference.

Maximum Seed Length

The `--ht-max-ext-seed-len` option limits the length of extended seeds populated into the hash table. Primary seeds that match many reference positions can be extended to achieve more unique matching, which may be required to map seeds within the maximum hit frequency (`--ht-max-seed-freq`). You can specify the length of primary seeds using `--ht-seed-len`.

Given a primary seed length k, the maximum seed length can be configured between k and k+128. The default is the upper bound, k+128.

Limit Seed Extension

The `--ht-max-ext-seed-len` option is recommended for short reads, such as reads less than 50 bp. For short reads, it can be helpful to limit seed extension to the read length minus a small margin, such as 1–4 bp. For example, with 36 bp reads, you could set `--ht-max-ext-seed-len` to 35. The setting makes sure that the hash table builder does not plan a seed extension longer than the read, which would cause seed extension and mapping to fail at run time for seeds that could have fit within the read with shorter extensions.

Seed extension can be similarly limited for longer reads. For example, by setting `--ht-max-ext-seed-len` to 99 for 100 bp reads. Limiting seed extension for longer reads does not provide as much utility because seeds are extended conservatively in any event. Even with the default $k+128$ limit, individual seeds are only extended to the lengths required to fit under the maximum hit frequency (`--ht-max-seed-freq`) and at most a few bases longer to approach the target hit frequency (`--ht-target-seed-freq`), or to avoid taking too many incremental extension steps.

Maximum Hit Frequency

The `--ht-max-seed-freq` option sets a limit on the number of seed hits (reference genome locations) that can be populated for any primary or extended seed. If a given primary seed maps to more reference positions than the specified limit, the primary seed must be extended long enough, so that the extended seeds subdivide into smaller groups of identical seeds under the limit. If, even at the maximum extended seed length (`--ht-max-ext-seed-len`), a group of identical reference seeds is larger than this limit, their reference positions are not populated into the hash table. Instead, DRAGEN populates a single high frequency record.

The maximum hit frequency can be configured from 1–256. If the value is too low, hash table construction can fail because too many seed extensions are needed. The recommended minimum for a whole human genome reference is 8.

Accuracy Considerations

A higher maximum hit frequency can lead to more successful mapping, due to the following.

- A higher limit rejects fewer reference positions that cannot map under it.
- A higher limit allows seed extensions to be shorter, which improves the odds of exact seed matching without overlapping variants or sequencing errors.

However, as with very short seeds, allowing high hit counts can sometimes lower mapping accuracy. Most of the seed hits in a large group are not to the true mapping location. Occasionally one of these noise hits could be reported due to imperfect scoring models. The mapper also limits the total number of reference positions considered. Allowing very high hit counts can potentially crowd out the actual best match from consideration.

Speed Considerations

Higher maximum hit frequencies slow down read mapping because seed mapping finds more reference locations, which result in additional work, such as Smith-Waterman alignments, to determine the best result.

Graph Reference

- `--ht-apply-graph`: Automatically apply population information for the auto-detected reference. The references which are auto-detected are hg19, hg38, hs37d5, and chm13. See TODO for more details on reference auto-detection.
- `--ht-pop-alt-contigs`: Population based alternate contigs FASTA.
- `--ht-pop-alt-liftover`: Liftover SAM file of population alternate contigs.
- `--ht-pop-snps`: Population based SNPs VCF

ALT-Contigs

The following options control See [References with ALT Contigs on page 16](#) for more information on building hash tables from a reference that contains ALT contigs.

- `--ht-mask-bed`: Set a custom BED file that defines which regions to mask. If not provided, the DRAGEN software automatically applies BED files for hg38 and hg19 from `/opt/edico/fasta_mask`.
- `--ht-alt-liftover`: Set a liftover file to build a liftover based ALT-aware hash table. SAM liftover files for hg38DH and hg19 are provided in `/opt/edico/liftover`.
- `--ht-allow-mask-and-liftover`: Allows you to use `--ht-mask-bed` and `--ht-alt-liftover` together.
- `--ht-suppress-mask`: Suppress automatic detection of the default mask bed files when building the hash table.

Decoy Contigs

- `--ht-decoys`: The DRAGEN software automatically detects the use of hg19 and hg38 references and adds decoys to the hash table when they are not found in the FASTA file. Use the `--ht-decoys` option to specify the path to a decoys file. The default is `/opt/edico/liftover/hs_decoys.fa`.
- `--ht-suppress-decoys`: Suppress automatic detection of the default decoys file when building the hash table.

Processing Options

- `--ht-num-threads`

The `--ht-num-threads` option determines the maximum number of worker CPU threads that are used to speed up hash table construction. The default for this option is 8, with a maximum of 32 threads allowed.

If your server supports execution of more threads, it is recommended that you use the maximum. For example, the DRAGEN Compute Server servers contain 24 cores that have hyperthreading enabled, so a value of 32 should be used. When using a higher value, adjust `--ht-max-table-chunks` needs to be adjusted as well. The servers have 128 GB of memory available.

- **`--ht-max-table-chunks`**

The `--ht-max-table-chunks` option controls the memory footprint during hash table construction by limiting the number of ~1 GB hash table chunks that reside in memory simultaneously. Each additional chunk consumes roughly twice its size (~2 GB) in system memory during construction.

The hash table is divided into power-of-two independent chunks, of a fixed chunk size, X, which depends on the hash table size, in the range $0.5 \text{ GB} < X \leq 1 \text{ GB}$. For example, a 24 GB hash table contains 32 independent 0.75 GB chunks that can be constructed by parallel threads with enough memory and a 16 GB hash table contains 16 independent 1 GB chunks.

The default is `--ht-max-table-chunks` equal to `--ht-num-threads`, but with a minimum default `--ht-max-table-chunks` of 8. These options match by default, because building one hash table chunk requires one chunk space in memory and one thread to work on it. However, there are build-speed advantages to raising `--ht-max-table-chunks` higher than `--ht-num-threads`, or to raising `--ht-num-threads` higher than `--ht-max-table-chunks`.

Size Options

- **`--ht-mem-limit`—Memory Limit**

The `--ht-mem-limit` option controls the generated hash table size by specifying the DRAGEN board memory available for both the hash table and the encoded reference genome. The `--ht-mem-limit` option defaults to 32 GB when the reference genome approaches WHG size, or to a generous size for smaller references. Normally there is little reason to override these defaults.

- **`--ht-size`—Hash Table Size**

This option specifies the hash table size to generate, rather than calculating an appropriate table size from the reference genome size and the available memory (`--ht-mem-limit`). Using default table sizing is recommended and using `--ht-mem-limit` secondary.

Seed Population Options

- **`--ht-ref-seed-interval`—Seed Interval**

The `--ht-ref-seed-interval` option defines the step size between positions of seeds in the reference genome populated into the hash table. An interval of 1 (default) means that every seed position is populated, 2 means 50% of positions are populated, and so on. Noninteger values are supported. For example, 2.5 yields 40% populated.

Seeds from a whole human reference can be 100% populated with 32 GB memory on DRAGEN boards. If using a substantially larger reference genome, modify this option accordingly.

- `--ht-soft-seed-freq-cap` and `--ht-max-dec-factor`—Soft Frequency Cap and Maximum Decimation Factor for Seed Thinning

Seed thinning is an experimental technique to improve mapping performance in high-frequency regions. When primary seeds have higher frequency than the cap indicated by the `--ht-soft-seed-freq-cap` option, only a fraction of seed positions are populated to stay under the cap. The `--ht-max-dec-factor` option specifies a maximum factor by which seeds can be thinned. For example, `--ht-max-dec-factor 3` retains at least 1/3 of the original seeds. `--ht-max-dec-factor 1` disables any thinning.

Seeds are decimated in careful patterns to prevent leaving any long gaps unpopulated. Seed thinning can achieve mapped seed coverage in high frequency reference regions where the maximum hit frequency would otherwise have been exceeded. Seed thinning can also keep seed extensions shorter, which can help improve successful mapping. Based on testing to date, seed thinning has not proven to be superior to other accuracy optimization methods.

- `--ht-rand-hit-hifreq` and `--ht-rand-hit-extend`—Random Sample Hit with HIFREQ Record and EXTEND Record

Whenever a HIFREQ or EXTEND record is populated into the hash table, the record stands in place of a large set of reference hits for a certain seed. Optionally, the hash table builder can choose a random representative of that set and populate that HIT record alongside the HIFREQ or EXTEND record.

Random sample hits provide alternative alignments that can help estimate MAPQ accurately for the alignments that are reported. Random sample hits are never used outside of this context for reporting alignment positions, because the random sample hits would result in biased coverage of locations that were selected during hash table construction.

To include a sample hit, set `--ht-rand-hit-hifreq` to 1. The `--ht-rand-hit-extend` option is a minimum preextension hit count to include a sample hit, or zero to disable. Modifying these options is not recommended.

Seed Extension Control

DRAGEN seed extension is dynamic, and applied as needed for particular K-mers that map to too many reference locations. Seeds are incrementally extended in steps of 2–14 bases (always even) from a primary seed length to a fully extended length. The bases are appended symmetrically in each extension step, which determines the next extension increment if any.

There is a potentially complex seed extension tree associated with each high frequency primary seed. Each full tree is generated during hash table construction and a path from the root is traced by iterative extension steps during seed mapping. The hash table builder employs a dynamic programming algorithm to search the space of all possible seed extension trees for an optimal one, by using a cost function that balances mapping accuracy and speed. The following options define that cost function:

- `--ht-target-seed-freq`—Target Hit Frequency

The `--ht-target-seed-freq` option defines the ideal number of hits per seed for which seed extension should aim. Higher values lead to fewer and shorter final seed extensions because shorter seeds tend to match more reference positions.

- `--ht-cost-coeff-seed-len`—Cost Coefficient for Seed Length

The `--ht-cost-coeff-seed-len` option assigns the cost component for each base by which a seed is extended. Additional bases are considered a cost because longer seeds risk overlapping variants or sequencing errors and losing their correct mappings. Higher values lead to shorter final seed extensions.

- `--ht-cost-coeff-seed-freq`—Cost Coefficient for Hit Frequency

The `--ht-cost-coeff-seed-freq` option assigns the cost component for the difference between the target hit frequency and the number of hits populated for a single seed. Higher values result primarily in high-frequency seeds being extended further to bring their frequencies down toward the target.

- `--ht-cost-penalty`—Cost Penalty for Seed Extension

The `--ht-cost-penalty` option assigns a flat cost for extending beyond the primary seed length. A higher value results in fewer seeds being extended at all. The default value is 0.

- `--ht-cost-penalty-incr`—Cost Increment for Extension Step

The `--ht-cost-penalty-incr` option assigns a recurring cost for each incremental seed extension step taken from primary to final extended seed length. More steps are considered a higher cost because extending in many small steps requires more hash table space for intermediate EXTEND records, which takes substantially more run time to execute the extensions. A higher value results in seed extension trees with fewer nodes that reach from the root primary seed length to leaf extended seed lengths in fewer, larger steps.

Pipeline Specific Hash Tables

RNA-Seq

When building a hash table, DRAGEN configures the options for DNA analysis by default. To run RNA-Seq data, you must build an RNA-Seq hash table by setting `--ht-build-rna-hashtable` to true. If running RNA-Seq alignment, use the original `--output-directory` instead of the automatically generated subdirectory.

CNV

If using the CNV pipeline, set `--enable-cnv` to true. The command generates an additional Kmer hash map that is used in the CNV algorithm. Illumina recommends to always use the `--enable-cnv` option, so you can perform CNV calling with the same hash table used for mapping and aligning.

Methylation

To run the methylation pipeline, you must build a methylation-specific hash table. DRAGEN can build a single-pass or legacy multi-pass methylation hash table. Methylation runs using a single-pass hash table are completed faster than the legacy multipass hash tables. Single-pass hash tables are recommended for building methylation tables and running analyses.

Hash Table Type	Hash Table Commands
single-pass	<ul style="list-style-type: none"> • --ht-methylated-combined=true • --ht-seed-len 27
multi-pass	<ul style="list-style-type: none"> • --ht-methylated=true • --ht-seed-len 27 • --ht-max-seed-freq 16

Single-Pass

The following is an example of a single-pass hash table build. The example generates a combined hash table in your reference index folder under the `methylConverted` subdirectory.

```
dragon --build-hash-table true \ --output-directory $REFDIR \ --ht-reference
$FASTA \ --ht-num-threads 40 \ --ht-methylated-combined=true \ --ht-seed-len 27
```

Multi-pass

Multi-pass methylation mapping require building two special hash tables with reference bases converted from C to T in one table and G to A in the other table. The conversions are performed automatically when using the `--ht-methylated` command line option. The converted hash tables are generated in two subdirectories under the folder specified using the `--output-directory` command line option. The subdirectories are named `CTConverted` and `GAConverted`, corresponding with the base conversions. When using the hash tables for methylated alignment runs, make sure to refer to the `--output-directory` folder, not the subdirectories.

The base conversions remove a significant amount of information from the hash tables. You might need to use different hash table parameters than in a conventional hash table build. The following options are recommended for building hash tables for mammalian species.

```
dragon --build-hash-table=true --output-directory $REFDIR --ht-reference $FASTA
--ht-max-seed-freq 16 --ht-seed-len 27 --ht-num-threads 40 --ht-methylated=true
```

HLA Hashtable

An HLA-specific anchored reference hashtable has to be created to run the HLA caller.

Set `--ht-build-hla-hashtable` to `true`. When the command runs, a `anchored_hla` subdirectory will be created inside the `--output-directory` directory. The HLA-specific reference subdirectory can be built at the same time as the primary reference construction.

An HLA resource file is packaged with DRAGEN and it is located at `/opt/edico/resources/hla/HLA_resource.v1.fasta.gz` after DRAGEN is installed. When building the HLA-specific anchored hash table, the HLA resource file will be used by default. A customized file can be used by using the `--ht-hla-reference` command. See [Using Custom HLA Reference Files](#) for more information.

Prepare a Custom Multigenome Reference

DRAGEN Bio-It Platform enables to build a custom multigenome hash table from a set of population variants (VCF). The method introduces alternate graph paths to the reference hash table to represent more broadly the allelic diversity of the population in specific regions defined in a graph-bed file. Gain on accuracy from this methodology has been described in scientific blogs available on the [Illumina Genomics Research Hub site](#).

In the end-to-end mode, the tool generates 3 intermediate files that are automatically given as input to the hash table builder to create a custom hash table using the ALT-liftover capability. These 3 intermediate files are:

- Isolated population SNPs (VCF): `pop_snps.vcf`
- Population alternative haplotypes (FASTA): `phased_alts.fasta`
- Population alternative alignments to the reference genome (SAM): `phased_alts.sam`

The tool also generates a `.csv` file `multigenome_reference_metrics.csv` to provide useful statistics information on the variants used to build the custom multigenome hash table.

The generated files, including `hash_table.cmp` and associated files in the specified output directory, can then be used as the reference hash table for the DRAGEN mapper.

The tool only supports human data and the following reference genome builds:

- CHM13
- hg38 (and related versions)
- hg19 (and related versions)
- hs37d5

Enable

Use the following command to enable the custom multigenome hash table builder:

```
dragen --build-hash-table true --ht-graph-vcf-list <path to a text file containing newline-separated phased VCF file names> --ht-reference <reference.fasta> --ht-graph-bed <graph.bed> --ht-mask-bed <mask.bed> --output-directory <DIR> [options]
```

Inputs

Set of population VCF

The custom multigenome hash table builder tool uses a set of population variants provided by the user to generate a multigenome hash table. The variants must be specified in VCF format, with one VCF file given per individual in the population (multisample VCF format is not permitted). These VCF files must have specific formatting described below.

The custom multigenome hash table builder tool only supports VCF file input respecting the format described below:

- single sample VCF supported, no gVCF
- with variants positionally sorted in the same contig order as the main FASTA reference genome provided in --ht-reference
- with the following FILTER codes, non-PASS records are ignored:
 - ##FILTER=<ID=PASS,Description="All filters passed">
- with the following FORMAT field, other FORMAT fields need to be removed:
 - ##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
 - ##FORMAT=<ID=PS,Number=1,Type=Integer,Description="Phase set identifier">

 | The INFO/FORMAT subfields must be defined in the header. Events with undefined subfields are ignored.

To build a high-performance custom genome it is highly recommended to use long read sequencing data. You can use an external tools, such as Whatshap, to generate phased input. DRAGEN leverage the phasing information to reconstruct population haplotypes.

Reference genome

A reference genome in FASTA format must be provided. DRAGEN only supports human reference genome builds: chm13, hg38, hg19 and hs37d5. Reference genomes are available to download from the [Illumina DRAGEN Bio-IT Platform Product Files](#) support page.

 | The reference genome provided as input must be from the same build as the one used to generate the input phased VCF's.

Graph bed file

A graph bed file must be provided in order to limit the positions to be included in the multigenome reference. Graph bed files are available to download from the [Illumina DRAGEN Bio-IT Platform Product Files](#) support page.

A custom graph bed file can also be provided given the following format: tab delimited with first three columns being: contig name, start position, end position. Any line with a contig name that is not found in the input FASTA is skipped. Any lines that describe empty intervals are skipped.

- i** Records of the graph bed file provided must be from the same build as the reference genome used to build the multigenome reference.

Mask bed file

A mask bed file must be provided to mask certain regions of high similarity between primary and alternate contigs present in the main genome FASTA. Mask bed files are available to download from the [Illumina DRAGEN Bio-IT Platform Product Files](#) support page.

A custom mask bed file can also be provided given the following format: tab delimited with first three columns being: contig name, start position, end position. Any line with a contig name that is not found in the input FASTA is skipped. Any lines that describe empty intervals are skipped.

- i** Records of the mask bed file provided must be from the same build as the reference genome used to build the multigenome reference.

Command Line Options

Option	Required	Description
--build-hash-table	Yes	Set to true. If it is set to false the pipeline will not use its end-to-end mode, and it will stop at the generation of the 3 intermediate files.
--ht-graph-vcf-list	Yes	Path to the text file containing the list of VCF files to be used to build the custom multigenome hash table
--ht-reference	Yes	Path to the reference genome FASTA file. Note: support only for human genome builds CHM13, hg38, hg19 and hs37d5.
--ht-graph-bed	Yes	Path to the graph bed file
--ht-mask-bed	Yes	Path to the mask bed file
--output-directory	Yes	Specify the directory where all related hash table files will be written

- i** The custom reference hash table pipeline will return an error if options `--ht-alt-liftover` or `-ht-allow-mask-and-liftover` are specified.

Determine Input and Output File Locations

The DRAGEN Bio-It Platform is very fast, which requires careful planning for the locations of the input and output files. If the input or output files are on a slow file system, then the overall performance of the system is limited by the throughput of that file system. It is recommended that inputs and outputs are streamed directly from/to a mounted external storage system.

The DRAGEN system is preconfigured with at least one fast file system consisting of a set of fast SSD disks grouped with RAID-0 for performance. This file system is mounted at `/staging`. This name was chosen to emphasize the fact that this area was built to be large and fast, but is not redundant. Failure of any of the file system's constituent disks leads to the loss of all data stored there.

During processing, DRAGEN generates and reads back temporary files. With DRAGEN, it is highly recommended to always direct temporary files to the fast SSD (or `/staging`) by using the `--intermediate-results-dir` option. If the `--intermediate-results-dir` option is not provided, temporary files are written to the `--output-directory`. DRAGEN recommends streaming inputs and outputs using an mounted external storage system.

Process Your Input Data

To analyze FASTQ data, use the `dragen` command. For example, the following command can be used to analyze a single-ended FASTQ file:

```
dragen \
-r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
-l /staging/test/data/SRA056922.fastq \
--output-directory /staging/test/output \
--output-file-prefix SRA056922_dragen \
--RGID DRAGEN_RGID \
--RGSM DRAGEM_RGSM
```

For more information on the command line options, see [DRAGEN Host Software on page 54](#).

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 \
--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 to 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/hg19/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 33.

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 \
-l /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 2 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 \
-l /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_cnvl.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 \
-l /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_hsxtens008_0123_FC543 with your own BCL directory.

- Enter the following input:

```
dragen --bcl-conversion-only=true \
--bcl-input-directory /mnt/san/131022_hsxtens008_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 \
-l 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
```

Multicaller Workflows

DRAGEN supports running multiple tools in a single workflow.

The *enable-component* flag controls which components are enabled or disabled. DRAGEN constructs a workflow using the enabled components and automatically resolves any component inconsistencies. When possible, DRAGEN runs components in parallel.

Each component has a set of options that configures input settings, internal algorithm parameters, or output files and filtering criteria. Refer to the individual component sections for more details.

As an example, a different BED file may be provided separately for each caller:

- cnv-target-bed
- sv-call-regions-bed
- vc-target-bed

Some options, such as *output-directory* and *sample-sex*, are shared amongst callers.

Each variant caller produces its own set of VCFs and metric output files.

Example Component Commands

```
enable-map-align
enable-sort
enable-duplicate-marking
enable-variant-caller
enable-cnv
enable-sv
```

Input Formats

DRAGEN accepts the following common standard NGS input formats:

- FASTQ (`fastq-file1` and `fastq-file2`)
- FASTQ List (`fastq-list`)
- BAM (`bam-input`)
- CRAM (`cram-input`)

Somatic workflows can use tumor equivalent input files (eg, `tumor-bam-input`).

When running from unaligned reads, the reads first go through the map/align component to produce alignments which continue downstream to the variant callers. When running from prealigned reads, DRAGEN supports re-aligning with the map/align component or using the existing alignments from the source input.

Multicaller Command Line Example

The following example demonstrates best practices for combining command line options from single caller scenarios to create a multicaller workflow. The example consists of the following steps:

- Configure the INPUT options.
- Configure the OUTPUT options.
- Configure MAP/ALIGN depending on if realignment is desired or not.
- Configure the VARIANT CALLERs based on the application.
- Build up the necessary options for each component separately, to enable reuse in the final command line.

The following are partial templates that can be used as starting points. Adjust them accordingly for your specific use case.

```
#!/bin/bash
set -euo pipefail
```

```
DRAGEN_HASH_TABLE=<REF_DIR>
FASTQ1=<fastq1>
FASTQ2=<fastq2>
RGSM=<RGSM>
RGID=<RGID>
OUTPUT=<OUT_DIR>
PREFIX=<OUT_PREFIX>

INPUT_OPTIONS=""
--ref-dir $DRAGEN_HASH_TABLE \
--fastq-file1 $FASTQ1 \
--fastq-file2 $FASTQ2 \
--RGSM $RGSM \
--RGID $RGID \
"

OUTPUT_OPTIONS=""
--output-directory $OUTPUT \
--output-file-prefix $PREFIX \
"

MA_OPTIONS=""
--enable-map-align true \
... <any other optional settings> \
"

CNV_OPTIONS=""
--enable-cnv true \
... <any other optional settings> \
"

SNV_OPTIONS=""
--enable-variant-caller true \
... <any other optional settings> \
"

SV_OPTIONS=""
--enable-sv true \
... <any other optional settings> \
"

CMD="
dragen \
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \
"
```

```
$MA_OPTIONS \
$CNV_OPTIONS \
$SNV_OPTIONS \
$SV_OPTIONS \
"
echo $CMD
bash -c $CMD
```

Germline

The following table summarizes the support for some input formats and variant callers. Some supported features and callers are not listed in the table.

GERMLINE	FASTQ with Map/Align	BAM/CRAM	BAM/CRAM with Map/Align
CNV + SNV	Supported	Supported	Supported
CNV + SV	Supported	Supported	Supported
SNV + SV	Supported	Supported	Supported
CNV + SNV + SV	Supported	Supported	Supported

Somatic

Somatic workflows specify both tumor and normal inputs. The need for potentially two input files (tumor and matched normal) as well as the need for a matched normal SNV VCF for the Somatic CNV caller means extra care has to be taken.

One recommended tumor/normal workflow first starts with running matched normal through Germline Workflow.

1. Run matched normal through Germline workflow (CNV + SNV + SV + ...). The workflow generates the matched normal SNV VCF.
2. Run tumor and matched normal through Somatic workflow (CNV + SNV + SV + ...)

Optionally, a full tumor/normal analysis can be done in a single execution if both the SNV and CNV modules are enabled, by leveraging the BAF information directly from the small variant caller. See the Somatic CNV section for more details. In brief, this requires the use of `--enable-variant-caller true` and `--cnv-use-somatic-vc-baf true`.

```
#!/bin/bash
set -euo pipefail

DRAGEN_HASH_TABLE=<REF_DIR>
```

```
TUMOR_BAM=<TUMOR_BAM>
NORMAL_BAM=<NORMAL_BAM>
OUTPUT=<OUT_DIR>
PREFIX=<OUT_PREFIX>

INPUT_OPTIONS=""
--ref-dir $DRAGEN_HASH_TABLE \
--tumor-bam-input $TUMOR_BAM \
--bam-input $NORMAL_BAM \
"

OUTPUT_OPTIONS=""
--output-directory $OUTPUT \
--output-file-prefix $PREFIX \
"

MA_OPTIONS=""
--enable-map-align false \
... <any other optional settings> \
"

CNV_OPTIONS=""
--enable-cnv true \
... <any other optional settings> \
"

SNV_OPTIONS=""
--enable-variant-caller true \
... <any other optional settings> \
"

SV_OPTIONS=""
--enable-sv true \
... <any other optional settings> \
"

CMD="
dragen \
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \
$MA_OPTIONS \
$CNV_OPTIONS \
$SNV_OPTIONS \
$SV_OPTIONS \
"
```

```
echo $CMD
bash -c $CMD
```

The following table lists the various combinations that are supported under the tumor/normal mode of operation.

Tumor normal	FASTQ with Map/Align	BAM/CRAM	BAM/CRAM with Map/Align
CNV + SNV	Supported	Supported	Not Supported
CNV + SV	Supported	Supported	Not Supported
SNV + SV	Supported	Supported	Not Supported
CNV + SNV + SV	Supported	Supported	Not Supported

To run in tumor only mode, remove the matched normal input from the INPUT options and configure each individual caller to run in tumor only mode. The following table lists the combinations that are supported under the tumor only mode of operation.

Tumor Only	FASTQ with Map/Align	BAM/CRAM	BAM/CRAM with Map/Align
CNV + SNV	Supported	Supported	Supported
CNV + SV	Supported	Supported	Supported
SNV + SV	Supported	Supported	Supported
CNV + SNV + SV	Supported	Supported	Supported

WES analysis is supported if the mode is supported in singe caller mode and there is no input configuration conflict.

For WES analysis, CNV requires a panel of normals regardless of whether it is Tumor Normal or Tumor Only analysis.

DRAGEN Host Software

You use the DRAGEN host software program *dragen* to build and load reference genomes, and then to analyze sequencing data by decompressing the data, mapping, aligning, sorting, duplicate marking with optional removal, and variant calling.

Invoke the software using the *dragen* command. The command-line options are described in the following sections.

Command-line options can also be set in a configuration file. For more information on configuration files, see [Configuration Files on page 74](#). If an option is set in the configuration file and is also specified on the command-line, the command-line option overrides the configuration file.

Command-line Options

For a complete list of command line options, see [Command Line Options on page 1](#).

DRAGEN Command-Line Options

Perform the following command-line options using the *dragen*:

- Build Reference/Hash Table

```
dragen --build-hash-table true --ht-reference <REF_FASTA> \
--output-directory <REF_DIRECTORY> [options]
```

- Run Map/Align and Variant Caller (*.fastq to *.vcf)

```
dragen -r <REF_DIRECTORY> --output-directory <OUT_DIRECTORY> \
--output-file-prefix <FILE_PREFIX> [options] -1 <FASTQ1> \
[-2 <FASTQ2>] --RGID <RG0> --RGSM <SM0> --enable-variant-caller true
```

- Run Map/Align (*.fastq to *.bam)

```
dragen -r <REF_DIRECTORY> --output-directory <OUT_DIRECTORY> \
--output-file-prefix <FILE_PREFIX> [options] \
-1 <FASTQ1> [-2 <FASTQ2>] \
--RGID <RG0> --RGSM
```

- Run Variant Caller Only (*.bam to *.vcf)

```
dragen -r <REF_DIRECTORY> --output-directory <OUT_DIRECTORY> \
--output-file-prefix <FILE_PREFIX> [options] -b <BAM> \
--enable-map-align false \
--enable-variant-caller true
```

- Re-map and Run Variant Caller (*.bam to *.vcf)

```
dragen -r <REF_DIRECTORY> --output-directory <OUT_DIRECTORY> \
--output-file-prefix <FILE_PREFIX> [options] -b <BAM> \
```

```
--enable-map-align true \
--enable-variant-caller true

• Run BCL Converter (BCL to *.fastq)
dragen --bcl-conversion-only true --bcl-input-directory <BCL_DIRECTORY> \
--output-directory <OUT_DIRECTORY>

• Run RNA Map/Align (*.fastq to *.bam)
dragen -r <REF_DIRECTORY> --output-directory <OUT_DIRECTORY> \
--output-file-prefix <FILE_PREFIX> [options] -1 <FASTQ1> \
[-2 <FASTQ2>] --enable-rna true
```

Reference Genome Options

Before you can use the DRAGEN system for aligning reads, you must load a reference genome and its associated hash tables onto the PCIe card. For information on preprocessing a reference genome's FASTA files into the native DRAGEN binary reference and hash table formats, see [Prepare a Reference Genome on page 11](#). You must also specify the directory containing the preprocessed binary reference and hash tables with the *-r* [or *--ref-dir*] option. This argument is always required.

Use the following command to load the reference genome and hash tables to DRAGEN card memory separately from processing reads.

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

Use the *-l* (*--force-load-reference*) option to force the reference genome to load even if it is already loaded.

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

The time needed to load the reference genome depends on the size of the reference, but for typical recommended settings, it takes approximately 30–60 seconds.

Preservation or Stripping of BQSR Tags

The Picard Base Quality Score Recalibration (BQSR) tool produces output BAM files that include tags BI and BD. BQSR calculates these tags relative to the exact sequence for a read. If a BAM file with BI and BD tags is used as input to mapper/aligner with hard clipping enabled, the BI and/or BD tags can become invalid.

The recommendation is to strip these tags when using BAM files as input. To remove the BI and BD tags, set the *--preserve-bqsr-tags* option to *false*. If you preserve the tags, DRAGEN warns you to disable hard clipping.

Read Group Options

DRAGEN assumes that all the reads in a given FASTQ belong to the same read group. DRAGEN creates a single @RG read group descriptor in the header of the output BAM file, with the ability to specify the following standard BAM attributes:

Attribute	Argument	Description
ID	--RGID	Read group identifier. If you include any of the read group parameters, RGID is required. It is the value written into each output BAM record.
LB	--RGLB	Library.
PL	--RGPL	Platform/technology used to produce the reads. The BAM standard allows for values CAPILLARY, LS454, ILLUMINA, SOLID, HELICOS, IONTORRENT and PACBIO.
PU	--RGPU	Platform unit, eg, flowcell-barcode.lane.
SM	--RGSM	Sample.
CN	--RGCN	Name of the sequencing center that produced the read.
DS	--RGDS	Description.
DT	--RGDT	Date the run was produced.
PI	--RGPI	Predicted mean insert size.

If any of these arguments are present, DRAGEN adds an RG tag to all the output records to indicate that they are members of a read group. The following example shows a command line that includes read group parameters:

```
dragen --RGID 1 --RGCN Broad --RGLB Solexa-135852 \
--RGPL Illumina --RGPU 1 --RGSM NA12878 \
-r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
-1 SRA056922.fastq --output-directory /staging/tmp/ \
--output-file-prefix rg_example
```

When using the `--fastq-list` option to input multiple read groups, BAM tags (and others) are specified for each read group by adding columns to the `fastq_list.csv` file. Each column heading consists of four capital letters and each begins with 'RG'. For each column, each read group's values for that column are propagated to the output BAM file in an identically named tag.

License Options

To suppress the license status message at the end of the run, use the `--lic-no-print` option. The following shows an example of the license status message:

```
LICENSE_MSG| =====
LICENSE_MSG| License report
LICENSE_MSG|   Genome status [ACxxxxxxxxxxxx] : used 1263.9 Gbases since 2018-
Feb-15 (1263886160894 bases, unlimited)
LICENSE_MSG|   Genome bases [ACxxxxxxxxxxxx] : 202000000
LICENSE_MSG|   Genome bases [total] : 202000000
```

Operating Modes

DRAGEN has two primary modes of operation, as follows:

- Mapper/aligner
- Variant caller

DRAGEN is capable of performing each mode independently or as an end-to-end solution. DRAGEN also allows you to enable and disable decompression, sorting, duplicate marking, and compression along the DRAGEN pipeline.

Full Pipeline Mode

To execute full pipeline mode, set `--enable-variant-caller` to `true` and provide input as unmapped reads in `*.fastq`, `*.bam`, or `*.cram` formats.

DRAGEN performs decompression, mapping, aligning, sorting, and optional duplicate marking and feeds directly into the variant caller to produce a VCF file. In this mode, DRAGEN uses parallel stages throughout the pipeline to drastically reduce the overall run time.

Map/Align Mode

Map/align mode is enabled by default. Input is unmapped reads in `*.fastq`, `*.bam`, or `*.cram` format.

DRAGEN produces an aligned and sorted BAM or CRAM file. To mark duplicate reads at the same time, set `--enable-duplicate-marking` to `true`.

Variant Caller Mode

To execute variant caller mode, set the `--enable-variant-caller` option to `true`, and set `--enable-map-align` option to `false`. The input must be a mapped and aligned with the BAM/CRAM file. DRAGEN produces a VCF file and will force-enable re-sorting of the BAM, because a number of read statistics and estimates are required for the Variant Caller to operate effectively. Setting `--enable-sort` to `false` will be overridden. BAM files cannot be duplicate marked in the DRAGEN pipeline prior to

variant calling if they have not already been marked. Use the end-to-end mode of operation to take advantage of the mark-duplicates feature.

RNA-Seq Data

To enable processing of RNA-Seq-based data, set `--enable-rna` to *true*.

DRAGEN uses the RNA spliced aligner during the mapper/aligner stage. DRAGEN dynamically switches between the required modes of operation.

Bisulfite MethylSeq Data

To enable processing of Bisulfite MethylSeq data, set the `--enable-methylation-calling` option to *true*. DRAGEN automates the processing of data for Lister (directional) and Cokus (nondirectional) protocols to generate a single BAM with bismark-compatible tags.

Alternatively, you can run DRAGEN in a mode that produces a separate BAM file for each combination of the C->T and G->A converted reads and references. To enable this mode of processing, you need to build a set of reference hash tables with `--ht-methylated` enabled, and run DRAGEN with the appropriate `--methylation-protocol` setting.

Output Options

The following command line options for output are mandatory:

- `--output-directory <out_dir>`—Specifies the output directory for generated files.
- `--output-file-prefix <out_prefix>`—Specifies the output file prefix. DRAGEN appends the appropriate file extension onto this prefix for each generated file.
- `-r [--ref-dir]`—Specifies the reference hash table.

The following examples do not include these mandatory options.

For mapping and aligning, the output is sorted and compressed into BAM format by default before saving to disk. You can control the output format from the map/align stage with the `--output-format <SAM/BAM/CRAM>` option. If the output file exists, DRAGEN issues a warning and exits. To force overwrite if the output file already exists, use the `-f [--force]` option.

For example, the following commands output to a compressed BAM file, and then forces overwrite:

```
dragen ... -f  
dragen ... -f --output-format bam
```

To generate a BAI-format BAM index file (*.bai), set `--enable-bam-indexing` to *true*.

The following example outputs to a SAM file, and then forces overwrite:

```
dragen ... -f --output-format sam
```

The following example outputs to a CRAM file, and then forces overwrite:

```
dragen ... -f --output-format cram
```

DRAGEN only outputs lossless CRAM files. All QNAMEs and BAM tags are preserved in the CRAM file.

Alignment tags

DRAGEN can generate mismatch difference (MD) tags, as described in the BAM standard. The feature is turned off by default because there is a small performance cost to generate these strings. To generate MD tags, set `--generate-md-tags` to true.

To generate ZS:Z alignment status tags, set `--generate-zs-tags` to true. These tags are only generated in the primary alignment and when a read has suboptimal alignments qualifying for secondary output (even if none were output because `--Aligner.sec-aligns` was set to 0). The following are valid tag values:

Tag	Tag meaning
ZS:Z:R	Multiple alignments with similar score were found.
ZS:Z:NM	No alignment was found.
ZS:Z:QL	An alignment was found but it was below the quality threshold.

To generate SA:Z tags, set `--generate-sa-tags` to true (the default). The tags provide alignment information (position, cigar, orientation) of groups of supplementary alignments, which are useful in structural variant calling.

To generate pair score in a ps:i tag, set `--generate-ps-tags` to true (false is the default for DNA, true is the default for RNA). The pair score is used in DRAGEN for computing MAPQ and can be used to check how well alignment candidates score against each other.

DRAGEN can also output mate alignment tags.

- To generate the mate cigar in the MC:Z tag, set `--generate-mc-tags` to true (true is the default).
- To generate the mate mapping quality in the MQ:i tag, set `--generate-mq-tags` to true (true is the default).
- To generate mate sequence in the R2:Z tag and mate base qualities in the Q2:Z tag, set `--generate-r2-tags` to true (false is the default), and set `--generate-q2-tags` to true (false is the default).

If enabled, R2:Z and Q2:Z tags are only emitted for improperly paired read alignments with fragment length's of at least 1,000 bp. Methylation pipelines do not support the output of mate alignment tags.

CRAM Output

When CRAM is selected as the output, DRAGEN generates a CRAM file with the following features:

- The CRAM file is created in V3.0 format
- The CRAM file is lossless. Lossy compression is not employed and is not an option

- The quality score compression is lossless and read names are preserved
- Only the GZIP compression algorithm is employed for maximum compatibility. Bgzip and Izma are not employed. RANS is used for quality scores
- All input BAM tags are preserved
- The DRAGEN Hash Table provided during the map/align run is the reference used to compress the CRAM file. When decompressing the CRAM file with a FASTA file and third-party tools, the FASTA file that was used to generate the Hash Table must be used.
- A CRAM index file is produced in a `.crai` format
- CRAM output is only possible when sort is enabled. CRAM alignments will always be positionally sorted

The following list of default settings are used for the CRAM output

CRAM Option	Value	Description
SEQS_PER_SLICE	1000	Max sequences per slice
BASES_PER_SLICE	SEQS_PER_SLICE*500	Max bases per slice
SLICE_PER_CNT	1	Max slices per container
embed_ref	0	Do not embed reference sequence
noref	0	Do not embed reference sequence
multiseq	-1	Do not use multiple references per slice
unsorted	0	Do not use unsorted mode
use_bz2	0	Do not compress using bzip2
use_Izma	0	Do not compress using Izma
use_rans	1	Use rANS for quality score compression
binning	NONE	Qual score binning not used
preserve_aux_order	1	Preserve all aux tags and order (incl RG,NM,MD)
preserve_aux_size	0	Aux tag sizes not preserved ('i', 's', 'c')
lossy_read_names	0	Preserve read names
lossy	0	Do not enable Illumina 8 quality-binning system
ignore_md5	0	Enable all checking of checksums
decode_md	0	Do not (re)generate MD and NM tags

Input Options

DRAGEN can process reads in FASTQ format or BAM/CRAM format. DRAGEN supports the following compression options for FASTQ input files.

- Uncompressed
- gzip or bgzip compression
- ORA compression. To use ORA compression, you must provide an ORA reference and reference directory. See [DRAGEN ORA Compression and Decompression](#) on page 668.

If your input FASTQ files are gzipped, automatically decompresses the files using hardware-accelerated decompression, and then streams the reads into the mapper. If your files end in *.ora, DRAGEN automatically decompresses the files using ORA decompression, and then streams the reads into the mapper. The same FASTQ command line options apply for all compression formats.

Input File Types

Use these command-line options for any input files you would like to use.

FASTQ Input Files

FASTQ input files can be single-ended or paired-end. Use the following examples to input FASTQ files.

- Single-ended in one FASTQ file (*-1 option*)

```
dragen -r <REF_DIR> -1 <fastq> --output-directory <OUT_DIR> \
--output-file-prefix <OUTPUT_PREFIX> --RGID <RGID> --RGSM <RGSM>
```

- Paired-end in two matched FASTQ files (*-1 options*) and (*-2 options*)

```
dragen -r <REF_DIR> -1 <fastq1> -2 <fastq2> \
--output-directory <OUT_DIR> --output-file-prefix <OUT_PREFIX> \
--RGID <RGID> --RGSM <RGSM>
```

- Paired-end in a single interleaved FASTQ file (*--interleaved (-i) option*)

```
dragen -r <REF_DIR> -1 <INTERLEAVED_FASTQ> -i \
--RGID <RGID> --RGSM <RGSM>
```

If using, bcl2fastq or the DRAGEN BCL command use the following common file naming convention:

```
<SampleID>_S<#>_<Lane>_<Read>_<segment#>.fastq.gz
```

Older versions of bcl2fastq and DRAGEN could segment FASTQ samples into multiple files to limit file size or to decrease the time to generate them.

For Example:

```
RDRS182520_S1_L001_R1_001.fastq.gz
RDRS182520_S1_L001_R1_002.fastq.gz
...
RDRS182520_S1_L001_R1_008.fastq.gz
```

The files above do not need to be concatenated to be processed together by DRAGEN. To map/align any sample, provide the first file in the series (-1<FileName>_001.fastq). DRAGEN reads all segment files in the sample consecutively for both of the FASTQ file sequences specified using the -1 and -2 options for paired-end input and for compressed fastq.gz files. To turn the behavior off, set --enable-auto-multifile to false on the command line.

DRAGEN can also optionally read multiple files by the sample name given in the file name, which can be used to combine samples that have been distributed across multiple BCL lanes or flow cells. To enable this feature, set the --combine-samples-by-name option to true.

If the FASTQ files specified on the command-line use the Casava 1.8 file naming convention shown above and additional files in the same directory share that sample name, those files and all their segments are processed automatically. Note that sample name, read number, and file extension must match. Index barcode and lane number can differ.

To avoid impacting system performance, input files must be located on a fast file system.

Multiple FASTQ Input Files

To provide multiple FASTQ input files, it is recommended to use the --fastq-list <csv file name> option to specify the name of a CSV file containing the list of FASTQ files, instead of using the --combine-samples-by-name option. For example:

```
dragen -r <ref_dir> --fastq-list <CSV_FILE> \
--fastq-list-sample-id <Sample_ID> \
--output-directory <OUT_DIR> --output-file-prefix <OUT_PREFIX>
```

Using a CSV file allows you to name the FASTQ input files, input from multiple subdirectories, and add BAM tags specified explicitly for each read group. DRAGEN automatically generates a CSV file of the correct format during BCL conversion to FASTQ. The CSV file is named `fastq_list.csv` and contains an entry for each FASTQ file or paired-end file pair produced during the run.

FASTQ CSV File Format

The first line of the CSV file specifies the title of each column, and is followed by one or more data lines. All lines in the CSV file must contain the same number of comma-separated alphanumeric values and should not contain blank spaces.

Column titles are case-sensitive. The following column titles are required:

- RGID—Read Group
- RGSM—Sample ID
- RGLB—Library
- Lane—Flow cell lane
- Read1File—Full path to a valid FASTQ input file
- Read2File—Full path to a valid FASTQ input file. Required for paired-end input. If not using paired-end input, leave empty.

Each FASTQ file can only be referenced once in the CSV list. All values in the Read2File column must be non-empty and reference valid files or must all be empty.

When generating a BAM file using fastq-list input, one read group is generated per unique RGID value. The BAM header contains RG tags for the following read groups:

- ID (from RGID)
- SM (from RGSM)
- LB (from RGLB)

You can specify additional tags for each read group by adding a column title. The column title must be only four upper-case alphanumeric characters and begin with RG. For example, to add a PU (platform unit) tag, add a column named RGPU and specify the value for each read group in this column. All column titles must be unique.

A fastq-list file can contain files for more than one sample. If a fastq-list file contains only one unique RGSM entry, then no additional options need to be specified and DRAGEN processes all files listed in the fastq-list file. If there is more than one unique RGSM entry in a fastq-list file, one of the following must also be specified in addition to `--fastq-list <filename>`.

- To process a specific sample from the CSV file, use `--fastq-list-sample-id <SampleID>`. Only the entries in the fastq-list file with an RGSM value that match the specified SampleID are processed.
- To process all samples together in the same run, regardless of the RGSM value, set `--fastq-list-all-samples` to true.

i For a single run, only one BAM and VCF output file are produced because all input read groups are expected to belong to the same sample. To process multiple samples from one BCL conversion run, run DRAGEN multiple times using different values for the `--fastq-list-sample-id` option.

There is no option to specify groupings or subsets of RGSM values for more complex filtering, but the fastq-list file can be modified to achieve the same effect.

The following is an example FASTQ list CSV file with the required columns:

```
RGID,RGSM,RGLB,Lane,Read1File,Read2File
CACACTGA.1,RDSR181520,UnknownLibrary,1,/staging/RDSR181520_S1_L001_R1_
001.fastq, /staging/RDSR181520_S1_L001_R2_001.fastq
AGAACCGGA.1,RDSR181521,UnknownLibrary,1,/staging/RDSR181521_S2_L001_R1_
001.fastq, /staging/RDSR181521_S2_L001_R2_001.fastq
TAAGTGCC.1,RDSR181522,UnknownLibrary,1,/staging/RDSR181522_S3_L001_R1_
001.fastq, /staging/RDSR181522_S3_L001_R2_001.fastq
AGACTGAG.1,RDSR181523,UnknownLibrary,1,/staging/RDSR181523_S4_L001_R1_
001.fastq, /staging/RDSR181523_S4_L001_R2_001.fastq
```

If you use the `--tumor-fastq-list` option for somatic input, use the `--tumor-fastq-list-sample-id <SampleID>` option to specify the sample ID for the corresponding FASTQ list, as shown in the following example:

```
dragen -r <ref_dir> --tumor-fastq-list <csv_file> \
--tumor-fastq-list-sample-id <Sample_ID> \
--output-directory <out_dir> \
--output-file-prefix <out_prefix> --fastq-list <csv_file_2> \
--fastq-list-sample-id <Sample_ID_2>
```

Tumor-Normal Pairs Input

In somatic mode, If using `fastq_lists` or `tumor_fastq_lists` that comprise multiple samples (RGSMs), you can use a loop to iterate through the two lists to create tumor-normal pairs for testing. Create a `*.txt` file with the RGSM of each normal sample to be tested (one per line), and then create a separate `*.txt` file with the RGSM of the tumor samples to be tested. Make sure that the tumor sample RGSM is listed in the same order as the corresponding normal samples and to include a blank line after the last sample.

You can use the following example script to perform testing in somatic mode. Each iteration takes one entry from the tumor samples list and one entry from the normal samples list (from top to bottom) to create a tumor-normal pair as input for the DRAGEN run.

```
#!/bin/bash
HT="/staging/HT/"
tumor_fastq_list="/staging/inputs/tumor_fastq_list.csv"
normal_fastq_list="/staging/inputs/normal_fastq_list.csv"
tumor_samples_list="/staging/inputs/tumor_samples_list.txt"
normal_samples_list="/staging/inputs/normal_samples_list.txt"
while read -u 3 -r tumor_RGSM && read -u 4 -r normal_RGSM; do
output_dir="/staging/results/${tumor_RGSM}_${normal_RGSM}"
mkdir -p ${output_dir}
```

```
dragen \
-r ${HT} \
--tumor-fastq-list ${tumor_fastq_list} \
--tumor-fastq-list-sample-id ${tumor_RGSM} \
--fastq-list ${normal_fastq_list} \
--fastq-list-sample-id ${normal_RGSM} \
--output-directory ${output_dir} \
--output-file-prefix ${tumor_RGSM}_${normal_RGSM}
done 3<${tumor_samples_list} 4<${normal_samples_list}
```

The following are examples of the FASTQ lists and samples lists used as input for the script.

Sample fastq_list.csv:

```
RGPL,RGID,RGSM,RGLB,Lane,Read1File,Read2File
DRAGEN_RGPL,DRAGEN_RGID_N1.1,normal-1,ILLUMINA,1,/staging/inputs/normal-1_S1_L001_R1_001.fastq.gz,/staging/inputs/normal-1_S1_L001_R2_001.fastq.gz
DRAGEN_RGPL,DRAGEN_RGID_N1.2,normal-1,ILLUMINA,2,/staging/inputs/normal-1_S1_L002_R1_001.fastq.gz,/staging/inputs/normal-1_S1_L002_R2_001.fastq.gz
DRAGEN_RGPL,DRAGEN_RGID_N2.1,normal-2,ILLUMINA,1,/staging/inputs/normal-2_S1_L001_R1_001.fastq.gz,/staging/inputs/normal-2_S1_L001_R2_001.fastq.gz
DRAGEN_RGPL,DRAGEN_RGID_N2.2,normal-2,ILLUMINA,2,/staging/inputs/normal-2_S1_L002_R1_001.fastq.gz,/staging/inputs/normal-2_S1_L002_R2_001.fastq.gz
DRAGEN_RGPL,DRAGEN_RGID_N3.1,normal-3,ILLUMINA,1,/staging/inputs/normal-3_S1_L001_R1_001.fastq.gz,/staging/inputs/normal-3_S1_L001_R2_001.fastq.gz
DRAGEN_RGPL,DRAGEN_RGID_N3.2,normal-3,ILLUMINA,2,/staging/inputs/normal-3_S1_L002_R1_001.fastq.gz,/staging/inputs/normal-3_S1_L002_R2_001.fastq.gz
```

Sample tumor_fastq_list.csv content:

```
RGPL,RGID,RGSM,RGLB,Lane,Read1File,Read2File
DRAGEN_RGPL,DRAGEN_RGID_T1.1,tumor-1,ILLUMINA,1,/staging/inputs/tumor-1_S1_L001_R1_001.fastq.gz,/staging/inputs/tumor-1_S1_L001_R2_001.fastq.gz
DRAGEN_RGPL,DRAGEN_RGID_T1.2,tumor-1,ILLUMINA,2,/staging/inputs/tumor-1_S1_L002_R1_001.fastq.gz,/staging/inputs/tumor-1_S1_L002_R2_001.fastq.gz
DRAGEN_RGPL,DRAGEN_RGID_T2.1,tumor-2,ILLUMINA,1,/staging/inputs/tumor-2_S1_L001_R1_001.fastq.gz,/staging/inputs/tumor-2_S1_L001_R2_001.fastq.gz
DRAGEN_RGPL,DRAGEN_RGID_T2.2,tumor-2,ILLUMINA,2,/staging/inputs/tumor-2_S1_L002_R1_001.fastq.gz,/staging/inputs/tumor-2_S1_L002_R2_001.fastq.gz
DRAGEN_RGPL,DRAGEN_RGID_T3.1,tumor-3,ILLUMINA,1,/staging/inputs/tumor-3_S1_L001_R1_001.fastq.gz,/staging/inputs/tumor-3_S1_L001_R2_001.fastq.gz
```

```
DRAGEN_RGPL,DRAGEN_RGID_T3.2,tumor-3,ILLUMINA,2,/staging/inputs/tumor-3_S1_L002_R1_001.fastq.gz,/staging/inputs/tumor-3_S1_L002_R2_001.fastq.gz
```

Sample normal_samples_list

```
normal-1
normal-2
normal-3
```

Sample tumor_samples_list content

```
tumor-1
tumor-2
tumor-3
```

FASTQ ORA Input Files

You can use the same options as the other FASTQ input file types for ORA files. To use the ORA file, replace the FASTQ file name with the ORA file name and specify the ORA reference directory using `--ora-reference`.

See [DRAGEN ORA Compression and Decompression on page 668](#) for more information on ORA reference files.

The following command represents paired-end in two matched ORA FASTQ files (-1 and -2 options).

```
dragen -r <REF_DIR> -1 <fastq.ora1> -2 <fastq.ora2> \
--ora-reference <ORADATA_DIR> \
--output-directory <OUT_DIR> --output-file-prefix <OUT_PREFIX> \
--RGID <RGID> --RGSM <RGSM>
```

BAM Input Files

BAM files can be used as input to the mapper/aligner. By default `--enable-map-align` is true. You can use the BAM file as input to the variant caller by setting the `--enable-map-align` option to false.

When you specify a BAM file as input, with map/align enabled, DRAGEN ignores any alignment information contained in the input file, and outputs new alignments for all reads.

If the input file contains paired-end reads, it is important to specify that the input data should be sorted so that pairs can be processed together. Other pipelines require you to resort the input data set by read name. DRAGEN vastly increases the speed of this operation by pairing the input reads, and sending on to the mapper/aligner when pairs are identified. Use the `--pair-by-name` option to enable or disable this feature (the default is true).

- Specify single-ended input in one BAM file with the **(-b)** and **--pair-by-name=false** options, as follows:

```
dragen -r <ref_dir> -b <bam> --output-directory <out_dir> \
--output-file-prefix <out_prefix> --pair-by-name false
```

- Specify paired-end input in one BAM file with the **(-b)** and **--pair-by-name=true** options, as follows:

```
dragen -r <ref_dir> -b <bam> --output-directory <out_dir> \
--output-file-prefix <out_prefix> --pair-by-name true
```

CRAM Input Files

You can use CRAM files as input to the DRAGEN mapper/aligner and variant caller. The DRAGEN functionality available when using CRAM input is the same as when using BAM input.

By default, the CRAM compressor and decompressor uses the DRAGEN reference specified with the **--ref-dir** option. CRAM compression is reference based, and the reference used for compression is not part of the CRAM file. Therefore, the CRAM input file must have been created with the same reference than what is provided to DRAGEN for the analysis.

DRAGEN supports the re-alignment of a CRAM input that was created with a different reference in one step. Re-aligning a CRAM file that was created with a different reference requires use of the **--cram-reference** option. This option will make the CRAM decompressor use the specified reference.

- cram-reference** can be either a fasta file, or a DRAGEN hash table folder.
- If pointing to a fasta file, the fasta .fai index file must be present next to the fasta file
- CRAM output will always be compressed using the **--ref-dir** reference

The following example was created with hg19, and re-analysis with hg38,

```
dragen -r <ref_dir HG38> --cram-input <cram> --output-directory <out_dir> \
--output-file-prefix <out_prefix> --cram-reference <ref_dir HG19>
```

```
dragen -r <ref_dir HG38> --cram-input <cram> --output-directory <out_dir> \
--output-file-prefix <out_prefix> --cram-reference <hg19.fa>
```

Use the following options to provide a CRAM input to either mapper/aligner or variant caller:

- cram-input**—The name and path for the CRAM file.
- cram-input**—One usage example is paired-end input in a single CRAM file. In addition, set the **--pair-by-name** option to true.

```
dragen -r <ref_dir> --cram-input <cram> --output-directory <out_dir> \
--output-file-prefix <out_prefix> --pair-by-name true
```

BCL Input Files

BCL is the output format of Illumina sequencing systems. Under limited circumstances, DRAGEN can read directly from BCL for map-align operations, saving the time needed for conversion to FASTQ.

DRAGEN can read directly from BCL in the following circumstances:

- Only one lane is input as part of a run (specified on the command-line).
- The lane has only a single sample specified in the SampleSheet.csv file.

When converting BCL to FASTQ is required, DRAGEN provides a BCL to FASTQ converter (see [BCL Data Conversion on page 613](#)).

The following example command is for BCL input with only one lane of input:

```
dragen --bcl-input-dir <BCL_ROOT> --bcl-only-lane <num> -r <ref_dir> \
--output-directory <out_dir> --output-file-prefix <out_prefix>
```

For additional BCL conversion options, see [Input File Types on page 61](#).

Handling of N bases

One of the techniques that DRAGEN uses to optimize handling sequences can lead to the overwriting the base quality score assigned to N base calls.

When you use the `--fastq-n-quality` and `--fastq-offset` options, the base quality scores are overwritten with a fixed base quality. The default values for these options are 2 and 33 to match the Illumina minimum quality of 35 (ASCII character #).

Read Names for Paired-End Reads

By a common convention, read names can include suffixes, such as /1 or /2, which indicate the end of a pair the read represents. For BAM input using the `--pair-by-name` option, DRAGEN ignores these suffixes to find matching pair names. By default, DRAGEN uses the forward slash character as the delimiter for these suffixes and ignores the /1 and /2 when comparing names. By default, DRAGEN strips these suffixes from the original read names.

DRAGEN provides the following options to control how suffixes are used:

- To change the delimiter character for suffixes, use the `--pair-suffix-delimiter` option. Valid values for this option include forward-slash (/), dot (.), and colon (:).
- To preserve the entire name, including the suffixes, set `--strip-input-qname-suffixes` to false.
- To append a new set of suffixes to all read names, set `--append-read-index-to-name` to true, where the delimiter is determined by the `--pair-suffix-delimiter` option. By default the delimiter is a slash, so /1 and /2 are added to the names.

Gene Annotation Input Files

When processing RNA-Seq data, you can supply a gene annotations file by using the `--annotation-file` option. Providing this file improves the accuracy of the mapping and aligning stage (see [Input Files on page 537](#)). The file should conform to the GTF/GFF format specification and should list annotated transcripts that match the reference genome being mapped against. The similar GFF3 format is currently not supported.

DRAGEN can take the `SJ.out.tab` file (see [SJ.out.tab on page 1](#)) as an annotations file to help guide the aligner in a two-pass mode of operation.

Networked Streaming

AWS S3, Azure Bob Storage, and HTTP Input Streaming

DRAGEN can stream input files directly from an AWS S3 bucket, an Azure blob, or using HTTP presigned URLs. You do not need to download the input files to a local disk prior to being processed. The files are streamed over the network directly into the DRAGEN processor.

Input streaming is most beneficial for large input files. DRAGEN supports input streaming for BAMs and compressed FASTQ files. For FASTQ files, input streaming can be used in all the configurations that use single-end FASTQs, paired-end FASTQs, and FASTQ lists.

Input streaming is supported for the following use cases.

- Mapping/aligning of FASTQ and BAM.
- Germline and somatic small variant calling from BAM (without remapping).

For other file types that are significantly smaller in size, download them locally before running the analysis.

Security and Permissions

To stream input files, you must have permission to access the remote files. 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.

Examples

The following examples display possible methods to stream input files directly with DRAGEN.

Streaming FASTQ Input Using AWS 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 Azure Storage Account

```
AZ_ACCOUNT_NAME="storage-account-name" AZ_ACCOUNT_KEY="" dragen -f
-r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
-1 https://storage-account-name.blob.core.windows.net/path/to/object_1.fastq.gz \
\
-2 https://storage-account-name.blob.core.windows.net/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
```

Streaming BAM Input Using AWS S3

```
dragen -f
-r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
-b s3://s3-bucket-name/path/to/object_1.bam \
--output-directory /staging/examples/ \
--output-file-prefix streaming
```

Streaming BAM Input Using Azure Storage Account

```
AZ_ACCOUNT_NAME="storage-account-name" AZ_ACCOUNT_KEY="" dragen -f
-r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
-b https://storage-account-name.blob.core.windows.net/path/to/object_1.bam \
--output-directory /staging/examples/ \
--output-file-prefix streaming
```

Streaming BAM Input Using HTTP

```
dragen -f
-r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \
-b https://bucket-name.amazonaws.com/path/to/object_1.bam?querystring \
--output-directory /staging/examples/ \
--output-file-prefix streaming
```

AWS S3, Azure Blob Storage, Output Streaming

DRAGEN can stream its output to an AWS S3 Bucket or an Azure Storage Account Container. Output streaming is beneficial for large output files and sharing results.

Streaming output to AWS S3

```
dragen -f
-r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149
-1 SRA056922.fastq
--RGID object_ID \
--RGSM sample_name \
--output-directory s3://s3-bucket-name/path/to/output \
--intermediate-results-dir /staging/examples \
--output-file-prefix streaming \
```

Streaming output to Azure Storage Account S3

```
AZ_ACCOUNT_NAME="storage-account-name" AZ_ACCOUNT_KEY="" dragen -f
-r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149
-1 SRA056922.fastq
--RGID object_ID \
--RGSM sample_name \
```

```
--output-directory https://storage-account-
name.blob.core.windows.net/path/to/output \
--intermediate-results-dir /staging/examples \
--output-file-prefix streaming \
```

Security and permissions

To stream input files or write to a cloud providers storage, you must have permission to access the remote files.

AWS S3

S3 requires AWS authentication and credentials. The authentication should already be set up on the instance you are running, for example, via IAM policies.

Azure Storage Account

Azure requires authentication and environment variables. DRAGEN supports two cases:

- Using managed identities
- Storage account access keys

To use managed identities you must run DRAGEN on an Azure instance. The instance must have `Contributor` permissions (read/write) on the Storage Account it wants to read and write to. If the instance has a single managed identity, only the `AZ_ACCOUNT_NAME=azure-storage-account-name` environment variable is required. For multiple managed identities, you must also provide the `AZR_IDENT_CLIENT_ID=<client-id>` environment variable, with the client id of the identity that can access your storage bucket. This can be found on the Azure Portal.

With storage account access keys, DRAGEN can write to an Azure bucket both on and off Azure instances. Find the Storage Account Access Key and set the environment variables `AZ_ACCOUNT_NAME=azure-storage-account-name` and `AZ_ACCOUNT_KEY=<account-key>`.

Presigned URL

An HTTP URL most likely has a query string attached to it, which provides the authentication credentials or necessary tokens to grant permission.

Sample Sex

Use the `--sample-sex` command line option to control the sex karyotype input used in downstream components, such as variant callers. If a sample sex karyotype input is not specified using the command line, the sex karyotype is automatically determined. The sex karyotype input is converted to a reference sex karyotype for use in variant calling. Other components might support sex karyotype input. Refer to the corresponding section for the component you are using.

The `--sample-sex` option supports the following values. Values are not case-sensitive.

- `none`—No sex karyotype input. Components use a default reference sex karyotype.
- `auto`—The sex karyotype is estimated by the Ploidy Estimator. If using CNV calling, sex karyotype is determined using a separate sex estimation module. If DRAGEN cannot estimate the sex karyotype, then components do not have a sex karyotype input. This behavior is then the same as `none`. `auto` is the default value.
- `female`—Sex karyotype input is XX.
- `male`—Sex karyotype input is XY.

The following example command lines use `--sample-sex` to specify the sex karyotype.

```
--sample-sex FEMALE
--sample-sex MALE
--sample-sex NONE
```

If the value is `none`, `female`, or `male`, the Ploidy Estimator could still run and produce output, but variant callers will not use any estimated sex karyotype that is different than the sex karyotype specified via the command-line.

The sex karyotype input is converted to the reference sex karyotype for the different components as follows. See the relevant component section for more information on how `--sample-sex` is used.

Sex	Small				
Karyotype Input	CNV Caller	ExpansionHunter	Ploidy Caller	Variant Caller	SV Caller
XX	XX	XX	XX	XX	XXYY
XY	XY	XY	XY	XY	XXYY
XXY	XY	XX	XY	XXYY	XXYY
XYY	XY	XY	XY	XXYY	XXYY
X0	XX	XY	XX	XXYY	XXYY
XXXY	XY	XX	XY	XXYY	XXYY
XXX	XX	XX	XX	XXYY	XXYY
None ¹	XX/XY	XX	XX	XXYY	XXYY

1. CNV independently checks the coverage ratio of X and Y to determine the reference sex karyotype. Detection of minimal Y coverage will yield XY, otherwise XX.

Autogenerated MD5SUM for BAM and CRAM Output Files

An MD5SUM file is generated automatically for BAM and CRAM output files. The MD5SUM file has the same name as the output file with an .md5sum extension appended (eg, whole_genome_run_123.bam.md5sum). The MD5SUM file is a single-line text file that contains the md5sum of the output file, which exactly matches the output of the Linux md5sum command.

The MD5SUM calculation is performed as the output file is written, so there is no measurable performance impact (compared to the Linux md5sum command, which can take several minutes for a 30x BAM).

Configuration Files

Command line options can be stored in a configuration file. The location of the default configuration file is /opt/edico/config/dragen-user-defaults.cfg. To specify a different file, use the --config-file (-c) option. The configuration file used for a given run supplies the default settings for that run. You can override any of the default settings using command line options.

DRAGEN recommends using the dragen-user-defaults.cfg as a template to create default settings for different use cases. Create your configuration file as follows.

1. Create a copy of dragen-user-defaults.cfg, and then rename the copy.
2. Modify the new configuration file according to your use case.

Make sure to modify your copy of dragen-user-defaults.cfg, not the original file.

DRAGEN recommends only adding options that do not change per run. Use the command line to specify options that vary per run.

Cloud Authentication and Licensing

Authentication is required for users that run DRAGEN on the cloud, with the Bring-Your-Own-License (BYOL) model, outside of integrated Illumina cloud products. A valid license is required to enable authentication and usage quotas.

License Server

DRAGEN cloud runs access the DRAGEN License Server to validate the credentials and licenses against the intended run. BYOL users must provide credentials and must allow access to the license server URL. The following command line option can be used to pass the credentials to DRAGEN: --lic-server=https://<user>:<pass>@license.edicogenome.com

An alternative way to provide license server credentials is by using a license credentials file. The `--lic-credentials input` command line option can be used to provide the full path to the license credentials file. It provides a more secure way to pass cloud credentials, which avoids accidental credentials leaks from command line console logs.

A license credentials file is a plain text file audited by the customer. The format is the same as the DRAGEN config files: = , each {key, value} separated by new line. The following key names must be used: `credentials1 = credentials2 =`

Instance Identity

DRAGEN cloud runs access the instance identity document via the Instance Metadata Service as part of the authentication. It uses the IPv4 local address. If access to the local address is not allowed, the authentication will fail. Alternately if the user does not want to allow applications to access this service, the user may save the instance identity document(s) and point DRAGEN to use them instead. The method for providing instance identity documents to DRAGEN is described below.

- Save the instance identity document(s) as files from the user's instance, and provide them as inputs to DRAGEN with each run
- The instance identity documents only need to be saved once per AWS account and region, and can be re-used subsequently

Examples for saving instance identity document(s):

AWS

```
curl -v -H Metadata:true --no-proxy "*" "http://169.254.169.254/latest/dynamic/instance-identity/pkcs7" -o /opt/instance-identity/pkcs7
curl -v -H Metadata:true --no-proxy "*" "http://169.254.169.254/latest/dynamic/instance-identity/document" -o /opt/instance-identity/document
cp /opt/instance-identity/pkcs7 /opt/instance-identity/signature
```

There should be 3 files in this folder, respectively named `pkcs7`, `signature` and `document`. Run DRAGEN using the `--lic-instance-id-location ${instance_identity}` command option.

Azure

```
curl -v -H Metadata:true --no-proxy "*" "http://169.254.169.254/metadata/instance?api-version=2020-09-01" -o /opt/instance-identity/instance
```

```
curl -v -H Metadata:true --noproxy "*" "http://169.254.169.254/metadata/attested/document?api-version=2020-09-01" -o /opt/instance-identity/document
```

There should be 2 files in this folder, respectively named `instance` and `document`. Run DRAGEN using the `--lic-instance-id-location ${instance_identity}` command option.

DRAGEN Reference Support

DRAGEN supports the construction of a reference hash tables for both human and non-human reference genomes. The reference autodetect feature of DRAGEN is able to recognize the reference hash tables build on the four Human reference genomes: hg19 (hg19), GRCh37/hs37d5 (hs37d5), GRCh38/hs38d1 (hg38), and T2T-CHM13v2.0 (chm13). Pre-built human reference hash tables are available for download at [Illumina DRAGEN Bio-IT Platform Support Site page](#).

DRAGEN also supports multigenome reference graphs hash tables which extend the reference genomes with alternative variant paths from a sample cohort used to construct the multigenome reference. A graph-based reference improves the mapping accuracy of Illumina reads in the “Difficult-to-Map Regions” of the genome and the downstream variant calling. Pre-built human multigenome reference graphs are available for download at [Illumina DRAGEN Bio-IT Platform Support Site page](#).

Refer to the following information, which summarize the reference support for each DRAGEN component and the recommended reference type for each component.

Germline

	Human hg19	Human hs37d5	Human hg38	Human chm13	Non- Human	Human Reference Type	Recommended Non-Human Reference Type
SNV	Yes	Yes	Yes	Yes	Yes	Graph	Non-Graph
CNV	Yes	Yes	Yes	Yes ¹	No	Graph	Non-Graph
SV	Yes	Yes	Yes	Yes ¹	Yes	Graph	Non-Graph
Expansion Hunter	Yes	Yes	Yes	No	No	Graph	Non-Graph
Targeted Callers	Yes	Yes	Yes	No	No	Graph	Non-Graph
RNA	Yes	Yes	Yes	Yes ¹	Yes	Non-Graph	Non-Graph
De Novo	Yes	Yes	Yes	Yes ¹	Yes	Graph	Non-Graph

	Human hg19	Human hs37d5	Human hg38	Human chm13	Non- Human	Human Reference Type	Recommended Non-Human Reference Type
Joint Genotyping	Yes	Yes	Yes	Yes ¹	Yes	Graph	Non-Graph
Biomarkers (HLA)	Yes	Yes	Yes	Yes ¹	No	Graph	Non-Graph
gVCF genotyper	Yes	Yes	Yes	Yes ¹	Yes	Graph	Non-Graph

1. The accuracy has not been established.

Somatic

	Human hg19	Human hs37d5	Human hg38	Human chm13	Non- Human	Recommended Human Reference Type	Recommended Non-Human Reference Type
SNV	Yes	Yes	Yes	Yes ¹	No	Non-Graph	Non-Graph
UMI SNV	Yes	Yes	Yes	Yes ¹	No	Non-Graph	Non-Graph
CNV	Yes	Yes	Yes	Yes ¹	No	Non-Graph	Non-Graph
SV	Yes	Yes	Yes	Yes ¹	No	Non-Graph	Non-Graph

1. The accuracy has not been established.

Methylation

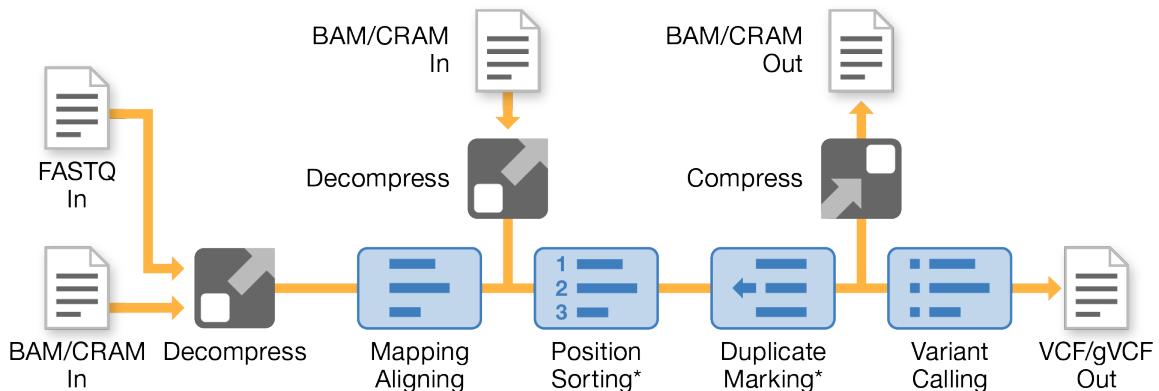
	Human hg19	Human hs37d5	Human hg38	Human chm13	Non- Human	Human Reference Type	Recommended Non-Human Reference Type
Methylation	Yes	Yes	Yes	No	No	Non-Graph	Non-Graph

Annotation

	Human hg19	Human hs37d5	Human hg38	Human chm13	Non- Human
Nirvana	Yes	Yes	Yes	No	Yes

DRAGEN DNA Pipeline

The DRAGEN DNA Pipeline accelerates the secondary analysis of NGS data by harnessing the tremendous power of the DRAGEN Bio-It Platform. The pipeline includes highly optimized algorithms for mapping, aligning, sorting, duplicate marking, and haplotype variant calling.



* Optional

DNA Mapping

DRAGEN primarily maps reads by finding exact reference matches to short seeds. However, DRAGEN can also map seeds differing from the reference by one nucleotide by also looking up single-SNP edited seeds. Seed editing is usually not necessary with longer reads (100 bp+), because longer reads have a high probability of containing at least one exact seed match. Seed editing is also not necessary if using paired ends, because a seed match from either mate can successfully align the pair. However, seed editing can be useful to increase mapping accuracy for short single-ended reads, with some cost in increased mapping time.

The following options control seed editing:

Table 3 Seed Editing Options

Command-Line Option Name	Configuration File Option Name
--Mapper.seed-density	seed-density
--Mapper.edit-mode	edit-mode
--Mapper.edit-seed-num	edit-seed-num
--Mapper.edit-read-len	edit-read-len

Command-Line Option Name	Configuration File Option Name
--Mapper.edit-chain-limit	edit-chain-limit

Seed Density

The `seed-density` option controls how many (normally overlapping) primary seeds from each read the mapper looks up in the hash table to find exact matches in the reference genome.

Seed density must be between 0.0 and 1.0. Internally, DRAGEN selects an available seed pattern equal to or close to the requested density. The most sparse pattern is one seed per 32 positions, or density 0.03125. The maximum density value of 1.0 generates a seed starting at every position in the read.

- Accuracy Considerations—Generally, denser seed lookup patterns improve mapping accuracy. However, for long reads (50 bp+) and low sequencing error rates, there is minimum improvement beyond the default 50% seed lookup density.
- Speed Considerations—Denser seed lookup patterns generally slow down mapping, while more sparse seed patterns speed up mapping. However, when the seed mapping stage runs faster than the aligning stage, a more sparse seed pattern does not improve the mapper speed.

Relationship to Reference Seed Interval

Functionally, a denser or more sparse seed lookup pattern has an affect similar to specifying a shorter or longer reference seed interval via the `--ht-ref-seed-interval` option. Populating 100% of reference seed positions and looking up 50% of read seed positions has the same effect as populating 50% of reference seed positions and looking up 100% of read seed positions. Either way, the expected density of seed hits is 50%.

More generally, the expected density of seed hits is the product of the reference seed density and the seed lookup density. For example, if 50% of reference seeds are populated and 33.3% (1/3) of read seed positions are looked up, then the expected seed hit density should be 16.7% (1/6).

Local Analysis Software automatically adjusts the seed lookup pattern to make sure it does not systematically miss the seed positions populated from the reference. For example, the mapper does not look up seeds matching only odd positions in the reference when only even positions are populated in the hash table, even if the reference seed interval is 2 and seed density is 0.5.

Edit Modes and Chain Limits

The `edit-mode` and `edit-chain-limit` options control when seed editing is used. The following four `edit-mode` values are available:

Mode Value	Description
0	No editing (default)
1	Chain length test
2	Paired chain length test
3	Full seed editing

Edit mode 0 requires all seeds to match exactly. Mode 3 is the most expensive because every seed that fails to match the reference exactly is edited.

Modes 1 and 2 employ heuristics to look up edited seeds only for reads most likely to be salvaged to accurate mapping. The main heuristic is a seed chain length test. Exact seeds are mapped to the reference in a first pass over a given read. The matching seeds are grouped into chains of similarly aligning seeds. If the longest seed chain in the read exceeds a threshold, `edit-chain-limit`, the read does not require seed editing, because there is already a mapping position.

Edit mode 1 triggers seed editing for a given read using the seed chain length test. If no seed chain exceeds `edit-chain-limit` or no exact seeds match, then a second seed-mapping pass is attempted using edited seeds.

Edit mode 2 further optimizes the heuristic for paired-end reads. If either mate has an exact seed chain longer than `edit-chain-limit`, then seed editing is disabled for the pair because a rescue scan is likely to recover the mate alignment based on seed matches from one read. Edit mode 2 is the same as mode 1 for single-end reads.

Seed Number and Read Length

For edit modes 1 and 2, when the heuristic triggers seed editing, the `edit-seed-num` and `edit-read-len` options control how many seed positions are edited in the second pass over the read. Although exact seed mapping can use a densely overlapping seed pattern, such as seeds starting at 50% or 100% of read positions, most of the value of seed editing can be obtained by editing a more sparse pattern of seeds, even a nonoverlapping pattern. Generally, if a user application can afford to spend some additional amount of mapping time on seed editing, you can obtain a greater increase in mapping accuracy for the same time cost by editing seeds in sparse patterns for many reads.

Whenever seed editing is triggered, these two options request `edit-seed-num` seed editing positions, distributed evenly over the first `edit-read-len` bases of the read. In an example with 21-base seeds where `edit-seed-num` is 6 and `edit-read-len` is 100, edited seeds can begin at offsets {0, 16, 32, 48, 64, 80} from the 5' end with consecutive seeds overlapping by five bases. When a particular read is shorter than `edit-read-len`, fewer seeds are edited.

Seed editing is more expensive when the `--ht-ref-seed-interval` option is greater than 1. For edit modes 1 and 2, additional seed editing positions are automatically generated to avoid missing the populated reference seed positions. For edit mode 3, the time cost can increase dramatically because query seeds matching unpopulated reference positions typically miss and trigger editing.

Map Orientations

The `--Mapper.map-orientations` option is used in mapping reads for bisulfite methylation analysis. The option is set automatically based on the value set for `--methylation-protocol`.

The `--Mapper.map-orientations` option can restrict the orientation of read mapping to only forward in the reference genome or only reverse-complemented. The following values are the valid values for `--map-orientations`:

- 0 is either orientation (default).
- 1 is only forward mapping.
- 2 is only reverse-complemented mapping.

If mapping orientations are restricted and paired end reads are used, the expected pair orientation can only be FR, not FF or RF.

DNA Aligning

Smith-Waterman Alignment Scoring Settings

The first stage of mapping generates seeds from the read and looks for exact matches in the reference genome. The seed match results are then refined by running full Smith-Waterman alignments on the locations with the highest density of seed matches. The alignment algorithm compares each position of the read against all candidate positions of the reference. These comparisons correspond to a matrix of potential alignments between read and reference. For each candidate alignment position, Smith-Waterman algorithm generates a score while passing through the scoring matrix. The score reflects if alignment is reached by a nucleotide match or mismatch, a deletion, or an insertion. A match between read and reference provides a bonus and a mismatch or indel imposes a penalty. The alignment chosen has the overall highest scoring path through the matrix.

For an alignment with multiple possible interpretations, the specific values chosen for scores indicate how to balance the possibility of an indel as opposed to one or more SNPs or the preference for an alignment without clipping. The default DRAGEN scoring values are reasonable for aligning moderate length reads to a whole human reference genome for variant calling applications. However, any set of Smith-Waterman scoring parameters represents an imprecise model of genomic mutation and sequencing errors. Differently tuned alignment scoring values can be more appropriate for some applications.

The following options control Smith-Waterman Alignment:

Option	Description
--Aligner.global	<p>The <code>global</code> option controls whether alignment is forced to be end-to-end in the read. The available values are 0 or 1.</p> <ul style="list-style-type: none"> When set to 1, alignments are always end-to-end, as in the Needleman-Wunsch global alignment algorithm. Alignments are not end-to-end in the reference. Alignment scores can be positive or negative. When set to 0, alignments can be clipped at either or both ends of the read, as in the Smith-Waterman local alignment algorithm, and alignment scores are positive. <p>For long reads, the value 0 is preferred, so significant read segments after a break can be clipped without severely decreasing the alignment score. Examples of breaks include a large indel, structural variant, chimeric read, and so forth. Setting the option to 1 might not have the desired effect with longer reads, because insertions at or near the ends of a read can function as pseudoclipping. Also, when <code>global</code> is 0, multiple (chimeric) alignments can be reported when various portions of a read match widely separated reference positions.</p> <p>For short reads, setting <code>global</code> to 1 is sometimes preferable. Short reads are unlikely to overlap structural breaks, unable to support chimeric alignments, and are suspected of incorrect mapping if they cannot align well end-to-end.</p> <p>To make a soft preference for unclipped alignments, consider using the <code>unclip-score</code> option, or increasing it, instead of setting <code>global</code> to 1.</p>
--Aligner.match-score	The <code>match-score</code> option specifies the score for a read nucleotide matching a reference nucleotide (A, C, G, or T), or matching a reference 2–3 nucleotide IUPAC-IUB code. The value is an unsigned integer from 0 to 15. Only set the <code>match_score</code> option to 0 when <code>global</code> is 1. A higher match score results in longer alignments and fewer long insertions.
--Aligner.match-n-score	The <code>match-n-score</code> option specifies the score for an aligned position where the read position and/or the reference position is an N code. The option is a signed integer from -16 to 15.

Option	Description
--Aligner.mismatch-pen	The mismatch-pen option sets the penalty, or negative score, for a read nucleotide mismatching any reference nucleotide or IUPAC-IUB code, except N. The option is an unsigned integer from 0 to 63. A higher mismatch penalty results in alignments with more insertions, deletions, and clipping to avoid SNPs.
--Aligner.gap-open-pen	The gap-open-pen option set the penalty, or negative score, for opening a gap (ie, an insertion or deletion). The value is only for a 0-base gap. The penalty is always added to the gap length multiplied by gap-ext-pen. The option is an unsigned integer from 0 to 127. A higher gap open penalty causes fewer insertions and deletions of any length in alignment CIGARs. Clipping or alignment through SNPs is used instead.
--Aligner.gap-ext-pen	The gap-ext-pen option sets the penalty, or negative score, for extending a gap (ie, an insertion or deletion) by one base. This option is an unsigned integer from 0 to 15. A higher gap extension penalty causes fewer long insertions and deletions in alignment CIGARs. Short indels, clipping, or alignment through SNPs is used instead.
--Aligner.unclip-score	The unclip-score option sets the score bonus for an alignment reaching the beginning or end of the read. A higher unclipped bonus causes alignment more often to reach the beginning and/or end of a read, where alignment can be done without too many SNPs or indels. An end-to-end alignment receives twice the bonus. The option is an unsigned integer from 0 to 127. A nonzero unclip-score is useful when global is 0 to make a soft preference for unclipped alignments. Unclipped bonuses have little effect on alignments when global is 1, because end-to-end alignments are forced. However, 2 × unclip-score does add to every alignment score unless no-unclip-score is 1). For longer reads, setting unclip-score much higher than gap-open-pen can result in insertions at or near one end of a read being used as pseudoclipping, as happens with global is 1.

Option	Description
--Aligner.no-unclip-score	<p>The <code>no-unclip-score</code> option can be 0 or 1. The default is 1. When <code>no-unclip-score</code> is set to 1, any unclipped bonus (<code>unclip-score</code>) contributing to an alignment is removed from the alignment score before further processing. Unclipped bonuses can include comparison with <code>aln-min-score</code>, comparison with other alignment scores, and reporting in AS or XS tags. However, the unclipped bonus still affects the best-scoring alignment found by the Smith-Waterman alignment to a given reference segment, biasing toward unclipped alignments.</p> <p>If a <code>unclip-score</code> greater than 0 causes a Smith-Waterman local alignment to extend out to one or both ends of the read, the following score changes are possible:</p> <ul style="list-style-type: none"> • If the <code>no-unclip-score</code> is 0, the alignment score stays the same or increases. • If <code>no-unclip-score</code> is 1, whereas it stays the same or decreases. <p>The default, <code>no-unclip-score</code> is 1, is recommended when <code>global</code> is 1, because every alignment is end-to-end. There is no need to add the same bonus to every alignment.</p> <p>When changing <code>no-unclip-score</code>, consider whether <code>aln-min-score</code> should be adjusted. When <code>no-unclip-score</code> is 0, unclipped bonuses are included in alignment scores compared to the <code>aln-min-score</code> floor, so the subset of alignments filtered out by <code>aln-min-score</code> can change significantly with <code>no-unclip-score</code>.</p>

Option	Description
--Aligner.aln-min-score	The <code>aln-min-score</code> option specifies a minimum acceptable alignment score. Any alignment results scoring lower are discarded. Increasing or decreasing <code>aln-min-score</code> can reduce or increase the percentage of reads mapped. This option is a signed integer (negative alignment scores are possible with <code>global=0</code>). <code>aln-min-score</code> also affects MAPQ estimates. The primary contributor to MAPQ calculation is the difference between the best and second-best alignment scores. A read's best alignment score is saved in the AS SAM tag, and the second-best score (if available) is saved in the XS tag. <code>aln-min-score</code> serves as the suboptimal alignment score if nothing higher was found except the best score. Therefore, increasing <code>aln-min-score</code> can decrease reported MAPQ for some low-scoring alignments. You can use the <code>min-score-coeff</code> option to adjust <code>aln-min-score</code> as a function of read length.
--Aligner.min-score-coeff	The <code>min-score-coeff</code> option makes adjustments to <code>aln-min-score</code> per read base. When using the <code>min-score-coeff</code> and <code>aln-min-score</code> options together, you can define the minimum alignment score for each read as an affine function of read length. The minimum score for an N-base read is calculated as follows. $(\text{min-score-coeff}) * N + (\text{aln-min-score})$ The <code>min-score-coeff</code> option is an integer ranging from -64 to 63.999. If the value is 0, then the minimum alignment score is fixed at <code>aln-min-score</code> for all read lengths. You can use positive values for <code>min-score-coeff</code> to allow shorter reads to match with lower alignment scores, but require longer reads to achieve higher scores.

Paired-End Options

DRAGEN can process paired-end data passed via a pair of FASTQ files or in a single interleaved FASTQ file. The hardware maps the two ends separately, and then determines a set of alignments that seem most likely to form a pair in the expected orientation and having roughly the expected insert size. The alignments for the two ends are evaluated for the quality of their pairing, with larger penalties for insert sizes far from the expected size.

The following options control processing of paired-end data:

Option	Description
--Aligner.pe-orientation	The pe-orientation option specifies the expected paired-end orientation. Only pairs with the orientation can be flagged as proper pairs. The following values are valid: <ul style="list-style-type: none"> • 0 is FR (default) • 1 is RF • 2 is FF
--Aligner.unpaired-pen	For paired-end reads, the best mapping positions are determined jointly for each pair. The positions are evaluated according to the largest pair score found, considering the various combinations of alignments for each mate. A pair score is the sum of the two alignment scores minus a pairing penalty, which estimates the unlikelihood of insert lengths further from the mean insert than this aligned pair. The unpaired-pen option specifies how much alignment pair scores should be penalized when the two alignments are not in properly paired position or orientation. The option also serves as the maximum pairing penalty for properly paired alignments with extreme insert lengths. The unpaired-pen option is specified in Phred scale, according to its potential impact on MAPQ. Internally, the penalty is scaled into the alignment score matrix based on Smith-Waterman scoring parameters.
--Aligner.pe-max-penalty	The pe-max-penalty option limits how much the estimated MAPQ for one read can increase because its mate aligned nearby. A paired alignment is never assigned MAPQ higher than the MAPQ that it would have received through single-end mapping, plus the value of pe-max-penalty. By default, pe-max-penalty = mapq-max = 255, effectively disabling this limit. The key difference between unpaired-pen and pe-max-penalty is that unpaired-pen affects calculated pair scores and thus which alignments are selected. The pe-max-penalty option affects only reported MAPQ for paired alignments.

Mean Insert Size Detection

When working with paired-end data, DRAGEN chooses likely pairs from the highest-quality alignments for the two ends. To make this choice, DRAGEN uses a skew normal insert model to evaluate the likelihood that a pair of alignments constitutes a pair. This model is based on the observation that

common library preparation methods have insert-size distributions that are sometimes close to normal, but also sometimes clearly asymmetric, often skewing toward longer insert sizes. The skew normal insert model is used only for the DNA mode.

If you know the statistics of the library prep for an input file and the file consists of a single read group, you can specify the following characteristics of the insert-length distribution:

- Mean
- Standard deviation
- Shape (or skewness)
- Three quartiles

These characteristics can be specified with the `Aligner.pe-stat-mean-insert`, `Aligner.pe-stat-stddev-insert`, `Aligner.pe-stat-shape-insert`, `Aligner.pe-stat-quartiles-insert`, and `Aligner.pe-stat-mean-read-len` options. However, allowing DRAGEN to detect these characteristics automatically is typically preferred.

To enable automatic sampling of the insert-length distribution, set `--enable-sampling` to true. When the software starts execution, it runs a sample of up to 100,000 pairs through the aligner, calculates the distribution, and then uses the resulting statistics for evaluating all pairs in the input set.

The DRAGEN host software reports the statistics in its stdout log, as follows:

```
Initial paired-end statistics detected for read group RGID, based on 39042 high
quality pairs for FR orientation
Quartiles (25 50 75) = 398 409 420
Mean = 410.192
Standard deviation = 14.1254
NOTE: DRAGEN's insert estimates include corrections for clipping (so they are
not identical to TLEN)
Skew-normal insert distribution applied:
Position (xi) = 424.084
Scale (omega) = 19.8719
Shape (alpha) = -1.88125
To rerun with identical insert stats, specify:
--Aligner.pe-stat-mean-insert=424.084
--Aligner.pe-stat-stddev-insert=19.8719
--Aligner.pe-stat-shape-insert=-1.88125
--Aligner.pe-stat-quartiles-insert="398 409 420"
--Aligner.pe-stat-mean-read-len=101
```

The Mean, Standard deviation, and Quartiles reported above are the sample mean, standard deviation and quartiles calculated from the initial sample of up to 100,000 pairs, assuming a normal distribution. The sample mean and standard deviation are used to fit the parameters of a skew-normal

distribution. A skew-normal distribution is defined by starting with an underlying normal distribution (whose mean we call `position` or `xi` and standard deviation we call `scale` or `omega`) and folding a varying portion of the probability mass from one side of the mean (e.g., left side) to the other (e.g., right) side. The portion folded varies smoothly, from 0% at the original mean, approaching 100% from the left tail to the right tail. A `shape` parameter which we call `alpha` controls how rapidly the folded fraction increases, and at `alpha=0` there is no folding and the distribution remains normal.

In the standard output, we also include the command line options needed to reproduce the DRAGEN run with the same insert stat settings. Note that when specifying stats on the command line, the skew-normal `xi` value should be used for `Aligner.pe-stat-mean-insert`. The `omega` value should be used for `Aligner.pe-stat-stddev-insert`, and the `alpha` value should be used for `Aligner.pe-stat-shape-insert`. If `Aligner-pe-stat-shape-insert` is not specified on the command line, a default value of 0 is assumed.

The insert length distribution for each sample is written to `fragment_length_hist.csv`. Each sample starts with the following lines:

```
#Sample: sample name
FragmentLength,Count
```

The lines are followed by the histogram.

When the number of sample pairs is very small, there is not enough information to characterize the distribution with high confidence. In this case, DRAGEN applies default statistics that specify a very wide insert distribution, which tends to admit pairs of alignments as proper pairs, even if they may lie tens of thousands of bases apart. In this situation, DRAGEN outputs a message, as follows.

```
WARNING: Less than 28 high quality pairs found - standard deviation is
calculated from the small samples formula
```

The small samples formula calculates standard deviation as follows:

```
if samples < 3 then
    standard deviation = 10000
else if samples < 28 then
    standard deviation = 25 * (standard deviation + 1) / (samples - 2)
end if
if standard deviation < 12 then
    standard deviation = 12
end if
```

The default model is standard deviation = 10000. If the first 100,000 reads are unmapped or all pairs are improper pairs, then the standard deviation is set to 10,000 and the mean and quartiles are set to 0. The minimum value for standard deviation is 12, which is independent of the number of samples. Also, in the DNA mode when we have fewer than 1000 high quality alignments we revert to the normal distribution based insert model, because of insufficient number of samples to accurately estimate the parameters of the skew normal distribution.

For RNA data, the insert size distribution is not normal due to pairs containing introns. The DRAGEN software estimates the distribution using a kernel density estimator to fit a long tail to the samples. The estimate leads to a more accurate mean and standard deviation for RNA data and proper pairing.

DRAGEN writes detected paired-end stats into a tab-delimited log file in the output directory called `.insert-stats.tab`. The file contains the statistical distribution of detected insert sizes for each read group, including quartiles, mean, standard deviation, shape minimum, and maximum. The information matches the standard output report. Also, the log file includes the minimum and maximum insert limits that DRAGEN applied for rescue scans.

The reported mean and standard deviation in this tab-limited log file are the `xi` and `omega` parameters of the skew-normal distribution.

Rescue Scans

For paired-end reads where a seed hit is found for one mate but not the other, rescue scans hunt for missing mate alignments within a rescue radius of the mean insert length. DRAGEN sets the default rescue radius to 2.5 standard deviations of the empirical insert distribution. In cases where the insert standard deviation is large compared to the read length, the rescue radius is restricted to limit mapping slowdowns. A warning message is displayed, as follows:

```
Rescue radius = 220
Effective rescue sigmas = 0.5
WARNING: Default rescue sigmas value of 2.5 was overridden by host
software!
The user may wish to set rescue sigmas value explicitly with --
Aligner.rescue-sigmas
```

Although the user can ignore this warning, or specify an intermediate rescue radius to maintain mapping speed, it is recommended to use 2.5 sigmas for the rescue radius to maintain mapping sensitivity. To disable rescue scanning, set `max-rescues` to 0.

Output Options

DRAGEN can track multiple independent alignments for each read. The alignments include the optimal (primary) one, alignments mapping different subsegments of the read (chimeric/supplementary), and suboptimal (secondary) mappings of the read to different areas of the reference.

For DNA alignment by default, DRAGEN can emit one primary alignment for each read, up to three chimeric alignments (`Aligner.supp-aligns=3`), and no secondary alignments (`Aligner.sec-aligns=0`). The maximum user-specified value for `supp-aligns` or `sec-aligns` is 4095.

The following configuration options control how many of each type of alignment to include in the DRAGEN output.

Option	Description
--Aligner.mapq-max	The <code>mapq-max</code> option specifies a limit on the estimated MAPQ that can be reported for any alignment. Values from 0 to 255 are valid. If the calculated MAPQ is higher, the <code>mapq-max</code> value is reported instead. The default is 60.
--Aligner.supp-aligns --Aligner.sec-aligns	The <code>supp-aligns</code> and <code>sec-aligns</code> options restrict the maximum number of supplementary (ie, chimeric and SAM FLAG 0x800) alignments and secondary (ie, suboptimal and SAM FLAG 0x100) alignments, respectively, that can be reported for each read. A maximum of 4095 supplementary alignments and 4095 secondary alignments can be reported for any read, in addition to a primary alignment. High settings for these two options impact speed so it is advisable to increase only as needed.
--Aligner.sec-phred-delta	The <code>sec-phred-delta</code> option controls the secondary alignments that are emitted based on the alignment score relative to the primary reported alignment. Only secondary alignments within this Phred value of the primary are reported.
--Aligner.sec-aligns-hard	The <code>sec-aligns-hard</code> option suppresses the output of all secondary alignments if there are more secondary alignments than can be emitted. When not all secondary alignments can be output, set <code>sec-aligns-hard</code> to 1 to force the read to be unmapped.
--Aligner.supp-as-sec	When the <code>supp-as-sec</code> option is set to 1, then supplementary (chimeric) alignments are reported with SAM FLAG 0x100 instead of 0x800. The default value is 0. The <code>supp-as-sec</code> option provides compatibility with tools that do not support FLAG 0x800.
--Aligner.hard-clips	The <code>hard-clips</code> option is used as a field of 3 bits, with values ranging from 0 to 7. The bits specify alignments, as follows. <ul style="list-style-type: none"> • Bit 0 is primary alignments • Bit 1 is supplementary alignments • Bit 2 is secondary alignments Each bit determines whether local alignments of that type are reported with hard clipping (1) or soft clipping (0). The default value is 6, meaning primary alignments use soft clipping and supplementary and secondary alignments use hard clipping.

DRAGEN Graph Mapper

The graph mapper in DRAGEN improves variant calling accuracy in segmental duplications and other regions that are difficult to map with Illumina reads. The graph-based method uses alt-aware mapping for population haplotypes that are stitched into the reference with known alignments. The method establishes alternate graph paths that reads can seed-map and align to. The graph mapper reduces mapping ambiguity because reads that contain population variants are attracted to the specific regions where the variants are observed.

To evolve the FASTA reference to a graph reference, DRAGEN augments the FASTA reference with around 900,000 short alternate contigs derived from population haplotypes of phased variants. The mapper has alt-aware capabilities that project reads that match the population haplotypes to corresponding primary assembly alignments with a precise lift-over alignment.

When given a set of population variants (VCF) or haplotypes, the FASTA modification is categorized in the following two types:

- Alternate contigs—This type represents population haplotypes. Alt-contigs can have a single variant or a combination of nearby phased variants.
- Ambiguous codes (IUPAC codes)—This type represents SNPs. To improve alignment, edit the reference FASTA with isolated population SNPs.

Read Trimming

DRAGEN can remove artifacts from reads using hardware accelerated read trimming. Hardware accelerated read trimming is available on U200 and cloud systems, as part of the DRAGEN mapper and adds no additional run time. DRAGEN provides multiple independent trimming filters that target different types of artifacts or use cases. You can enable and configure the artifacts or use cases independently to tailor the read-trimming to your analysis. Read trimming uses two different modes, hard-trimming and soft-trimming.

To enable hard-trimming mode, use `--read-trimmers`. In hard-trimming mode, potential artifacts are removed from input reads. Reads that are trimmed to fewer than 20 bases are filtered and replaced with a placeholder read that uses 10 N bases. DRAGEN assigns the filtered reads a 0x200 flag set.

DRAGEN contains a novel lossless soft-trimming mode. In soft-trimming mode, reads are mapped as though they had been trimmed, but no bases are removed. To enable the trimmer in soft mode, use `--soft-read-trimmers`.

Soft-trimming suppresses systematic mismapping of reads that contain trimmable artifacts, without actually losing the trimmed bases in aligned output. Soft-trimming prevents reads with trimmable artifacts, such as Poly-G artifacts, from being mapped to reference G homopolymers, or prevents adapter sequences from being mapped to the matching reference loci. Soft-trimming might map reads to different positions in the reference than they would have been if not using soft-trimming. When using soft-trimmed, DRAGEN does not filter reads and does not map reads with bases that would have been trimmed entirely.

Soft-trimming for Poly-G artifacts is enabled by default on supported systems.

Read Trimming Tools

Fixed Length Trimming

Fixed-length trimming removes a fixed number of bases from the 5' end of each read. If you are analyzing sequencing data from an amplicon of fixed size and expect the read-length to consistently exceed the length of quality sequence data, you can use the expected number in fixed-length trimming.

Poly-G Trimming

Poly-G artifacts appear on two-channel sequencing systems when the dark base G is called after synthesis has terminated. As a result, DRAGEN calls several erroneous high-confidence G bases on the ends of affected reads. For contaminated samples, many affected reads can be mapped to reference regions with high G content. The affected reads can cause problems for processing downstream.

Quality Trimming

Base quality can degrade over the length of a read toward the 5' end and separate from any artifacts from early termination of synthesis. The lower quality bases can affect mapping and alignment results, and might lead to incorrect variant or methylation calls downstream. The quality trimming tool calculates a rolling average of the base quality inward from the 5' end and removes the minimum number of bases, so the average number of bases is above the threshold specified using `--trim-min-quality`.

Adapter Trimming

Problems during library preparation, or libraries with smaller inserts can result in the synthesis of high quality reads containing sequence from the adapters used. If not removed before analysis, noninsert bases can reduce mapping efficiency and downstream accuracy. The adapter trimming tool uses the adapter sequences from the input FASTA file, and then removes all hits greater than a specified size. Adapter trimming allows for a 10% mismatch. For 3' adapters, trimming is from the first matching adapter base to the end of the read. For 5' adapters, trimming is from the first (3') matching adapter base to the beginning (5') of the read.

Ambiguous Base Trimming

If quality trimming is not feasible due to reduced yield or other limitations, an alternative option is to remove only explicitly ambiguous bases from the ends of read. If enabled the ambiguous base trimmer applies a simple exact-match search to both ends of all processed reads, regardless of mate-pair status.

Minimum Length Trimming

You can maximize trimmer sensitivity, by using the minimum length trimming tool to remove a fixed number of bases from each read after the trimmer tools above have run. For example, if you would like to remove 5 bp from each read, a 7 bp adapter hit could be missed if five of the bases are removed first. To mitigate this issue, DRAGEN provides an optional minimum trim-length filter.

Maximum Length Trimming

If using libraries of fixed-size inserts, such as small PCR amplicons, it is more convenient to specify a length that all reads should be trimmed to rather than the number of bases to remove. You can use the maximum length trimming tool.

PolyA Tail Trimming

If using RNA libraries, reads overlapping the poly-A tail of the transcripts may contain long poly-A/poly-T sequences at the end of the reads which may result in incorrect alignment. The poly-A trimmer mitigates this by trimming the poly-A tail from the end of the read. See [PolyA Trimming](#) for more information..

Read Trimming Metrics

The trimmer generates a metrics file titled `<output_prefix>.trimmer_metrics.csv`. Metrics are available on an aggregate level over all input data. The metrics units are in reads or bases.

Metric	Description
Total input reads	Total number of reads in the input files.
Total input bases	Total number of bases in the input reads.
Total input bases R1	Total number of bases in R1 reads.
Total input bases R2	Total number of bases in R2 reads.
Average input read length	Total number of input bases divided by the number of input reads.
Total trimmed reads	Total number of reads trimmed by at least one base, not including soft-trimming.
Total trimmed bases	Total number of bases trimmed, not including soft-trimming.
Average bases trimmed per read	The number of trimmed bases divided by the number of input reads.

Metric	Description
Average bases trimmed per trimmed read	The number of trimmed bases divided by the number of trimmed reads.
Remaining poly-G K-mers R1 3prime	The number of R1 3' read ends that contain likely Poly-G artifacts after trimming.
Remaining poly-G K-mers R2 3prime	The number of R2 3' read ends that contain likely Poly-G artifacts after trimming.
Total filtered reads	The number of reads that were filtered out during trimming.
Reads filtered for minimum read length R1	The number of R1 reads that were filtered out due to being trimmed below the minimum read length.
Reads filtered for minimum read length R2	The number of R2 reads that were filtered out due to being trimmed below the minimum read length.
<Trimmer tool> trimmed reads	The number of reads with at least one base trimmed by TRIMMER. DRAGEN reports the metric for both R1 and R2 mates and the filtering status (unfiltered or filtered) of the trimmed read. The metric includes reads that were trimmed during soft-trimming. Each trimming tool above produces the metric.
<Trimmer tool> trimmed bases	The number of bases trimmed by TRIMMER. The metric is produced for both R1 and R2 mates and the filtering status (unfiltered or filtered) of the trimmed read. The metric includes bases from reads that were trimmed during soft trimming. Each trimming tool above produces the metric.

Read Trimming Settings

Read Trimmer

Metric	Description
--read-trimmers	<p>To enable trimming filters in hard-trimming mode, set to a comma-separated list of the trimmer tools you would like to use. To disable trimming, set to <code>none</code>. During mapping, artifacts are removed from all reads.</p> <p>Read trimming is disabled by default.</p> <p>The following are valid trimmer names:</p> <ul style="list-style-type: none"> <code>fixed-len</code>—Fixed-length trimming <code>polyg</code>—Poly-G trimming <code>quality</code>—Quality trimming <code>adapter</code>—Adapter trimming <code>n</code>—Ambiguous base trimming <code>min-len</code>—Minimum length trimming <code>cut-end</code>—Maximum length trimming <code>polya</code>—RNA Poly-A tail trimming. See PolyA Trimming for more information. <code>bisulfite</code>—Bisulfite trimming
--soft-read-trimmers	<p>To enable trimming filters in soft-trimming mode, set to a comma-separated list of the trimmer tools you would like to use. To disable soft trimming, set to <code>none</code>. During mapping, reads are aligned as if trimmed, and bases are not removed from the reads.</p> <p>Soft-trimming is enabled for the <code>polyg</code> filter by default.</p> <p>The following are the valid trimmer names:</p> <ul style="list-style-type: none"> <code>fixed-len</code>—Fixed-length trimming <code>polyg</code>—Poly-G trimming <code>quality</code>—Quality trimming <code>adapter</code>—Adapter trimming <code>n</code>—Ambiguous base trimming <code>min-len</code>—Minimum length trimming <code>cut-end</code>—Maximum length trimming <code>polya</code>—RNA Poly-A tail trimming. See PolyA Trimming for more information. <code>bisulfite</code>—Bisulfite trimming
--trimming-only	Disables mapping and alignment to run read-trimming only.

Filter after Trimming Execution

Option	Description
--trim-min-length	Specify a minimum read length allowed after the trimmer execution. DRAGEN filters any reads with a length less than the value after all read-trimming steps are completed (default: 20).
--trim-min-len-read1	Specify a minimum read length allowed for read1 after the trimmer execution. DRAGEN filters any reads with a length of read1 less than the value after all read-trimming steps are completed (default: 20).
--trim-min-len-read2	Specify a minimum read length allowed for read2 after the trimmer execution. DRAGEN filters any reads with a length of read2 less than the value after all read-trimming steps are completed (default: 20).
--trim-filter-dummy-len	Specify the number of N bases in dummy reads that replace filtered reads (default: 10).
--trim-filter-set-flag	If enabled, dummy reads will have their 0x200 SAM flag set (default: true).

Fixed Length Trimming

Option	Description
--trim-r1-5prime	Specify a fixed number of bases to trim from the 5' end of Read 1 (default: 0).
--trim-r1-3prime	Specify a fixed number of bases to trim from the 3' end of Read 1 (default: 0).
--trim-r2-5prime	Specify a fixed number of bases to trim from the 5' end of Read 2 (default: 0).
--trim-r2-3prime	Specify a fixed number of bases to trim from the 3' end of Read 2 (default: 0).

Quality Trimming

Option	Description
--trim-min-quality	Specify the minimum read quality. DRAGEN trims bases from the 3' end of reads with a quality below the value.
--trim-r1-5prime	Specify a fixed number of bases to trim from the 5' end of Read 1 (default: 0).
--trim-r1-3prime	Specify a fixed number of bases to trim from the 3' end of Read 1 (default: 0).
--trim-r2-5prime	Specify a fixed number of bases to trim from the 5' end of Read 2 (default: 0).
--trim-r2-3prime	Specify a fixed number of bases to trim from the 3' end of Read 2 (default: 0).

Adapter Trimming

Option	Description
--trim-adapter-read1	Specify the FASTA file that contains adapter sequences to trim from the 3' end of Read 1.
--trim-adapter-read2	Specify the FASTA file that contains adapter sequences to trim from the 3' end of Read 2.
--trim-adapter-r1-5prime	Specify the FASTA file that contains adapter sequences to trim from the 5' end of Read 1. NB: the sequences should be in reverse order (with respect to their appearance in the FASTQ) but not complemented.
--trim-adapter-r2-5prime	Specify the FASTA file that contains adapter sequences to trim from the 5' end of Read 2. NB: the sequences should be in reverse order (with respect to their appearance in the FASTQ) but not complemented.
--trim-adapter-stringency	Specify the minimum number of adapter bases required for trimming (default: 4).

Bisulfite Trimming

Option	Description
--trim-bisulfite-ends	Enable both 5'-Prime and 3'-Prime bisulfite trimming.
--trim-bisulfite-5prime	If a 3' adapter was trimmed, trim an additional 2bp from the 3' end, unless the 5' end matches 'CAA' or 'CGA'.
--trim-bisulfite-3prime	If the 5' end matches 'CAA' or 'CGA', trim the first two of these 5' bases.

Minimum Length Trimming

Option	Description
--trim-min-r1-5prime	Specify the minimum number of bases to trim from the 5' end of Read 1 (default: 0).
--trim-min-r1-3prime	Specify the minimum number of bases to trim from the 3' end of Read 1 (default: 0).
--trim-min-r2-5prime	Specify the minimum number of bases to trim from the 5' end of Read 2 (default: 0).
--trim-min-r2-3prime	Specify the minimum number of bases to trim from the 3' end of Read 2 (default: 0).

Maximum Length Trimming

Option	Description
--trim-max-length	Specify the maximum number of bases that can be trimmed from the sequences of both reads.
--trim-max-len-read1	Specify the maximum number of bases that can be trimmed from the sequences of read1.
--trim-max-len-read2	Specify the maximum number of bases that can be trimmed from the sequences of read2.

PolyA Trimming

Option	Description
--trim-polya-min-trim	The minimum number of poly-As required for polyA trimming (default: 20).

PolyG Trimming

Option	Description
--trim-polyg-kmer-len	How many bases to check at each read end for poly-G artifact detection (default: 25).
--trim-polyg-kmer-non-g	The maximum number of non-G bases in the K-mer for poly-G artifact detection (default: 2).
--trim-polyg-g-score-r1-5prime	The score for G bases on the 5' end of read 1 (default: 0).
--trim-polyg-g-score-r1-3prime	The score for G bases on the 3' end of read 1 (default: 15).
--trim-polyg-g-score-r2-5prime	The score for G bases on the 5' end of read 2 (default: 0).
--trim-polyg-g-score-r2-3prime	The score for G bases on the 3' end of read 2 (default: 15).
--trim-polyg-min-trim-r1-5prime	The minimum number of G's to trim from the 5' end of read 1 (default: 6).
--trim-polyg-min-trim-r1-3prime	The minimum number of G's to trim from the 3' end of read 1 (default: 6).
--trim-polyg-min-trim-r2-5prime	The minimum number of G's to trim from the 5' end of read 2 (default: 6).
--trim-polyg-min-trim-r2-3prime	The minimum number of G's to trim from the 3' end of read 2 (default: 6).
--trim-polyg-early-exit-threshold	The signed score threshold for poly-G trimming to exit early (default: -500).

PolyX Trimming

Option	Description
--trim-polyx-bases-r1-5prime	The bases to trim for polyX trimming from the 5' end of read 1 (default: empty string "").
--trim-polyx-bases-r1-3prime	The bases to trim for polyX trimming from the 3' end of read 1 (default: empty string "").
--trim-polyx-bases-r2-5prime	The bases to trim for polyX trimming from the 5' end of read 2 (default: empty string "").
--trim-polyx-bases-r2-3prime	The bases to trim for polyX trimming from the 3' end of read 2 (default: empty string "").
--trim-polyx-min-trim-r1-5prime	The minimum number of X's to trim from the 5' end of read 1 (default: 20).
--trim-polyx-min-trim-r1-3prime	The minimum number of X's to trim from the 3' end of read 1 (default: 20).
--trim-polyx-min-trim-r2-5prime	The minimum number of X's to trim from the 5' end of read 2 (default: 20).
--trim-polyx-min-trim-r2-3prime	The minimum number of X's to trim from the 3' end of read 2 (default: 20).

DRAGEN FastQC

The DRAGEN FastQC module is a tool for calculating common metrics used for quality control of high-throughput sequencing data. The tool is modeled after the metrics generated by the FastQC tool from Babraham Institute.

The metrics are generated automatically on all DRAGEN map-align workflows, with no additional run time, and output in a CSV format file called <PREFIX>.fastqc_metrics.csv. All metrics are calculated and reported separately for each mate-pair.

If you are only interested in sample QC or would like to obtain FastQC results only, DRAGEN provides a mode to generate the `fastqc_metrics.csv` file directly.

By default DRAGEN FastQC and read-trimming are run as preprocessing steps to standard sequence alignment workflows. If DNA alignment is not needed, or if QC results are needed more quickly, the mapping and BAM output portions of the workflow can be disabled. The workflow only outputs key metric files and runs ~70% faster. To use this option, enter `--fastqc-only=true` to the DRAGEN command.

i If FastQC runs stand-alone, then the license will not be consumed.

If FastQC runs with map-align enabled, then the license will be consumed.

Differences from the Babraham Institutes' FastQC

DRAGEN's FastQC module is a complete reimplementation of the original FastQC tool developed by the Babraham Institute (BI-FastQC). The reimplementation of FastQC in DRAGEN has been modified to take advantage of the hardware-acceleration provided by DRAGEN's Field-Programmable Gate Array (FPGA) for a significant speed improvement. There are some differences in how the values are calculated and the resulting metrics will not be identical between the two tools. The most significant differences are described below.

- Binning: BI-FastQC uses a customizable binning strategy with a default of 5bp bins, while DRAGEN uses an algorithmic binning strategy based on the Granularity setting resulting in DRAGEN providing more precise results at the default settings.
- Outputs: BI-FastQC text output contain the same information as their plots in tabular format, while DRAGEN-FastQC outputs its raw data. For example, BI-FastQC both plots an outputs the average base quality per-position, while DRAGEN outputs the average base quality by both position and nucleotide. This allows for a more detailed analysis of the data, but requires slightly more work to generate the associated plot.
- Rounding: DRAGEN consistently rounds its calculations to the nearest integer, while the original FastQC uses a mixture of rounding and taking the mathematical floor, leading DRAGEN-FastQC to provide incrementally higher results for some metrics.
- Smoothing: Both DRAGEN-FastQC and BI-FastQC utilize smoothing techniques for their distributions of %GC, to account for the fact that 151bp do not divide evenly into 100 percentile bins. To take advantage of the speed offered by the FPGA, DRAGEN utilizes a slightly different algorithm than BI-FastQC which results in slightly different results.

Metric Granularity

Due to memory constraints, it is not possible to guarantee single-base resolution for all metrics.

DRAGEN provides an algorithmic solution for binning via `--fastqc-granularity`. DRAGEN allocates 256 bins in memory for each size or position-based metric. The granularity value of 4–7 inclusive can be used to determine the bin size. High values use smaller bins for greater resolution. Lower values can be used to create larger bins for larger read-lengths.

Granularity	Single Base Resolution (bp)	Resolution at 150 (bp)	Recommended Read-Lengths (bp)
7 (default)	1–255	1	< 256
6	1–128	2	≥ 256 and < 507

Granularity	Single Base Resolution (bp)	Resolution at 150 (bp)	Recommended Read-Lengths (bp)
5	1–64	4	≥507 and < 4031
4	1–32	8	≥4031

Adapter and Kmer Sequence Files

To include metrics for adapter or other sequence content, DRAGEN FastQC needs the desired sequences to be provided in FASTA format. For this purpose, DRAGEN provides the following options for this purpose:

- For adapter sequences, use `--fastqc-adapter-file`.
- For any additional kmers of interest, use `--fastqc-kmer-file`.

With the `--fastqc-kmer-file` option, you can add sequences of interest without changing the expected adapter results.

DRAGEN FastQC can accept up to a combined total of 16 adapters and kmer sequences. Each sequence can be a maximum of 12 bp in length. By default, DRAGEN uses the adapter file located at `/opt/edico/config/adapter_sequences.fasta`. The file contains the following adapter sequences, which are the same as the FastQC from the Babraham Institute (v 0.11.10 and later).

- Illumina Universal Adapter—AGATCGGAAGAG
- Illumina Small RNA 3' Adapter—TGGAATTCTCGG
- Illumina Small RNA 5' Adapter—GATCGTCGGACT
- Nextera Transposase Sequence—CTGTCTCTTATA

FastQC Metrics Output

The FastQC metrics are output to a CSV file format in the run output directory called `<PREFIX>.fastqc_metrics.csv`.

The reported metrics are organized into eight sections by metric type. Each section is categorized into separate rows by length, position, or other relevant categorical variables. The following metric types compose the sections.

Option	Description
Read Mean Quality	Total number of reads. Each average Phred-scale quality value is rounded to the nearest integer.

Option	Description
Positional Base Mean Quality	Average Phred-scale quality value of bases with a specific nucleotide and at a given location in the read. Locations are listed first and can be either specific positions or ranges. The nucleotide is listed second and can be A, C, G, or T. N or ambiguous bases are assumed to have the system default value, usually QV2.
Positional Base Content	Number of bases of each specific nucleotide at given locations in the read. Locations are given first and can be either specific positions or ranges. The nucleotide is listed second and can be A, C, G, T, N.
Read Lengths	Total number of reads with each observed length. Lengths can be either specific sizes or ranges, depending on the settings specified using --fastqc-granularity.
Read GC Content	Total number of reads with each GC content percentile between 0% and 100%.
Read GC Content Quality	Average Phred-scale read mean quality for reads with each GC content percentile between 0% and 100%.
Sequence Positions	Number of times an adapter or other kmer sequence is found, starting at a given position in the input reads. Sequences are listed first in the metric description in quotes. Locations are listed second and can be either specific positions or ranges.
Positional Quality	Phred-scale quality value for bases at a given location and a given quantile of the distribution. Locations are listed first and can be either specific positions or ranges. Quantiles are listed second and can be any whole integer 0–100.

The following examples include rows from each section.

Section	Mate	Metric	Value
READ MEAN QUALITY	Read1	Q38 Reads	965377
POSITIONAL BASE MEAN QUALITY	Read1	ReadPos 145- 152 T Average Quality	34.49
POSITIONAL BASE MEAN QUALITY	Read1	ReadPos 150 T Average Quality	34.44
POSITIONAL BASE MEAN QUALITY	Read1	ReadPos 256+ T Average Quality	36.99
POSITIONAL BASE CONTENT	Read1	ReadPos 145- 152 A Bases	113362306

Section	Mate	Metric	Value
POSITIONAL BASE CONTENT	Read1	ReadPos 150 A Bases	14300589
POSITIONAL BASE CONTENT	Read1	ReadPos 256+ A Bases	13249068
READ LENGTHS	Read1	150bp Length Reads	77304421
READ LENGTHS	Read1	144-151bp Length Reads	77304421
READ LENGTHS	Read1	>=255bp Length Reads	1000000
READ GC CONTENT	Read1	50% GC Reads	140878674373
READ GC CONTENT QUALITY	Read1	50% GC Reads Average Quality	36.20
SEQUENCE POSITIONS	Read1	'AGATCGGAAGAG' 137bp Starts	20
SEQUENCE POSITIONS	Read1	'AGATCGGAAGAG' 137-144bp Starts	23
POSITIONAL QUALITY	Read1	ReadPos 150 50% Quantile QV	37
POSITIONAL QUALITY	Read1	ReadPos 145-152 50% Quantile QV	37

Mapping with ALT-Aware

The GRCh38 human reference contains many more alternate haplotypes, or ALT contigs, than previous versions of the reference. Generally, including ALT contigs in the mapping reference improves mapping and variant calling specificity, because misalignments are eliminated for reads matching an ALT contig but scoring poorly against the primary assembly. However, mapping with GRCh38's ALT contigs without special treatment can substantially degrade variant calling sensitivity in corresponding regions, because many reads align equally well to an ALT contig and to the corresponding position in the primary assembly.

Masked Based ALT-awareness

The recommended and default approach for dealing with ALT-contigs in DRAGEN is masking regions of ALT contigs of high similarity to their corresponding primary contig. This approach is more accurate than liftover based ALT-awareness because there are many places where the "correct" or most useful liftover between a long ALT haplotype and the primary assembly is ambiguous. Incorrect liftover can produce dense clusters of mismapped reads and false variant calls. The base masking approach has the benefits of using ALT contigs without the negative consequences.

Masked hash tables are built from a standard hg18 or hg38 FASTA that contains ALT contigs. The hash table builder will automatically mask regions of the ALT contigs with Ns.

Liftover Based ALT-awareness

With liftover based ALT-awareness, the mapper and aligner are aware of the liftover relationship between ALT contig positions and corresponding primary assembly positions. Seed matches within ALT contigs are used to obtain corresponding primary assembly alignments, even if the latter score poorly. Liftover groups are formed, each containing a primary assembly alignment candidate, and zero or more ALT alignment candidates that lift to the same location. Each liftover group is scored according to its best-matching alignments, taking properly paired alignments into account. The winning liftover group provides its primary assembly representative as the primary output alignment, with MAPQ calculated based on the score difference to the second-best liftover group. Emitting primary alignments within the primary assembly maintains normal aligned coverage and facilitates variant calling there. If the --Aligner.en-alt-hap-aln option is set to 1 and --Aligner.supp-aligns is greater than 0, then corresponding alternate haplotype alignments can also be output, flagged as supplementary alignments.

If ALT contigs are detected in an hg19 or GRCh38 reference, DRAGEN requires ALT-Aware hash tables. To disable this requirement in DRAGEN, set the --ht-alt-aware-validate option to false.

The following is a comparison of alternative approaches for dealing with alternate haplotypes.

- Mapping without ALT contigs in the reference:
 - Reads matching ALT contigs can misalign and result in a false-positive variant call.
 - Poor mapping and variant calling sensitivity where reads matching an ALT contig differ greatly from the primary assembly.
- Mapping with ALT contigs but no ALT awareness:
 - Reads matching ALT contigs do not misalign and related false-positive variant calls are prevented.
 - Low or zero aligned coverage in primary assembly regions covered by alternate haplotypes, because some reads are mapping to ALT contigs.
 - Low or zero MAPQ in regions covered by alternate haplotypes, where they are similar or identical to the primary assembly.
 - Variant calling sensitivity is reduced throughout regions covered by alternate haplotypes.
- Mapping with ALT contigs and alt awareness:
 - Reads matching ALT contigs do not misalign and related false-positive variant calls are prevented.
 - Normal aligned coverage in regions covered by alternate haplotypes, because primary alignments are mapping to the primary assembly.
 - Normal MAPQs are assigned because alignment candidates within a liftover group are not considered in competition.
 - Good mapping and variant calling sensitivity where reads matching an ALT contig differ greatly from the primary assembly.

Sorting

The map/align system produces a BAM file sorted by reference sequence and position by default. Creating this BAM file typically eliminates the requirement to run `samtools sort` or any equivalent postprocessing command. To enable or disable the creation of the BAM file, you can use the `--enable-sort` option as follows.

- To enable, set to true.
- To disable, set to false.

On the reference hardware system, running with sort enabled increases the run time for a 30x full genome by about 6–7 minutes.

Filter Duplicate Variants

DRAGEN can find and remove variants that are common to separate VCF files. DRAGEN supports the following modes:

- Small indel deduplication—if using a structural variant VCF and a small variant VCF, DRAGEN filters all small indels in the structural variant VCF that appear and are passing in the small variant VCF (PASS in the FILTER column of the small variant VCF file). You must provide a reference genome to generate the VCF files to normalize the variants. DRAGEN normalizes variants by trimming and left shifting by up to 500 bases. An instance of utilizing this feature is when incorporating both SV and SNV callers in somatic workflows, which can increase sensitivity and prevent the occurrence of replicated variants within genes such as FLT3 and KMT2A.
- SMN deduplication—if using a small variant VCF and an ExpansionHunter VCF, DRAGEN filters any lines in the small variant VCF that have the same chromosome and position as lines in the ExpansionHunter VCF with the INFO tag `VARID=SMN`. A reference genome is not required.

Use the following command line options to input VCF or gVCF files. The input files are not altered.

- `vd-sv-vcf`—Specify a structural variant VCF or gVCF.
- `vd-small-variant-vcf`—Specify a small variant VCF or gVCF.
- `vd-eh-vcf`—Specify an ExpansionHunter VCF or gVCF.

DRAGEN determines the name and type of the output file as follows.

Component	Description
Output prefix	If a value is specified for <code>output-file-prefix</code> , the prefix is used as usual. If the value is not valid, the name of the filtered input is used as the prefix.

Component	Description
Deduplication mode	The prefix is followed by <code>.small_indel_dedup</code> or <code>.smn_dedup</code> depending on the deduplication mode used.
File type	The output file type matches the input file type (VCF or gVCF). If <code>enable-vcf-compression</code> is set to true, the output file is gzip compressed, regardless of if the input file was compressed.

Command-Line Options

You can use the following command line options for variant deduplication.

Option	Description
<code>enable-variant-deduplication</code>	To enable variant deduplication, set to <code>true</code> . The default is <code>false</code> .
<code>enable-vcf-indexing</code>	To generate tabix index files, set to <code>true</code> . The default is <code>true</code> .
<code>vd-output-match-log</code>	To log matching lines to a text file, set to <code>true</code> . The default is <code>false</code> . For each match, the two matching lines follow each other, then by a new line. The name of the match log is either <code>match_log.smn_dedup.txt</code> or <code>match_log.small_indel_dedup.txt</code> depending on which deduplication mode you use.

The following is an example command for an SMN deduplication standalone run:

```
dragen --enable-map-align false \
--enable-variant-deduplication true \
--vd-small-variant-vcf <small variant vcf> \
--vd-eh-vcf <expansion hunter vcf> \
--output-directory /tmp/ \
--vd-output-match-log true \
```

You can also run small indel deduplication automatically on outputs from the DRAGEN joint caller where both structural variant and small variant callers are enabled. To run small indel deduplication automatically, set `enable-variant-deduplication` to `true`, and make sure the `vd-sv-vcf`, `vd-small-indel-vcf`, and `vd-eh-vcf` input options are not set. Only small indel deduplication can be run automatically.

The following is an example command for an automatic small indel deduplication run.

```
dragen \
--ref-dir <REF>
--output-directory <DIR> \
```

```
--output-file-prefix <PREFIX> \
-b <BAM> \
--enable-map-align false \
--enable-variant-caller true \
--enable-sv true \
--enable-variant-deduplication true \
--vd-output-match-log true \
```

Duplicate Marking

Marking or removing duplicate aligned reads is a common best practice in whole-genome sequencing. Not doing so can bias variant calling and lead to incorrect results.

The DRAGEN system can mark or remove duplicate reads and produce a BAM file with duplicates either marked in the FLAG field or entirely removed.

Algorithm

The DRAGEN duplicate-marking algorithm is modeled on the `MarkDuplicates` feature from the Picard toolkit. All the aligned reads are grouped into subsets. All the members of each subset are potential duplicates.

For two pairs to be duplicates, they must meet the following requirements:

- Identical alignment coordinates at both ends. Position is adjusted for soft- or hard-clips from the CIGAR.
- Identical orientations from the direction of the two ends, with the left-most coordinate being first.

If an unpaired read has an identical coordinate and orientation to either end of any other read, whether paired or not, it can be marked as a duplicate.

Unmapped or read pairs are never marked as duplicates.

DRAGEN identifies a group of duplicates, selects the best duplicate of the group, and marks the others with the BAM PCR or optical duplicate flag (0x400 or decimal 1024). For the comparison, duplicates are scored based on the average sequence Phred quality. Pairs receive the sum of the scores of both ends, while unpaired reads get the score of the one mapped end. The score is intended to preserve the reads with the highest-quality base calls.

If two pairs (or reads) have exactly matching quality scores, DRAGEN breaks the tie by choosing the pair with the higher alignment score. If there are multiple pairs that also tie on alignment score, then DRAGEN chooses a pair arbitrarily.

The score for an unpaired read R is the average Phred quality score per base, calculated as follows:

Where:

- R is a BAM record.

- `QUAL` is its array of Phred quality scores.
- `dedup-min-qual` is a DRAGEN configuration option with default value of 15.

For a pair, the score is the sum of the scores for the two ends.

The score is stored as a one-byte number, with values rounded down to the nearest one-quarter. Rounding can lead to different duplicate marks than the ones chosen by Picard. However, the impact on variant calling results is negligible because the reads are close in quality.

Limitations

The following limitations apply to DRAGEN duplicate marking implementation:

- When there are two duplicate reads or pairs with close Phred sequence quality scores, DRAGEN might choose a different winner from the one chosen by Picard. The differences have negligible impact on variant calling results.
- If using a single FASTQ file as input, DRAGEN accepts only a single library ID as a command-line argument (`PGLB`). For this reason, the FASTQ inputs to the system must already be separated by library ID. Library ID cannot be used as a criterion for distinguishing nonduplicates.

Settings

The following options can be used to configure duplicate marking in DRAGEN:

Option	Description
<code>--enable-duplicate-marking</code>	To enable duplicate marking, set to true. When <code>--enable-duplicate-marking</code> is enabled, the output is sorted, regardless of the value of the <code>enable-sort</code> option.
<code>--remove-duplicates</code>	To suppress the output of duplicate records, set to true. If set to false, set the 0x400 flag in the FLAG field of duplicate BAM records. When <code>--remove-duplicates</code> is enabled, then <code>enable-duplicate-marking</code> is enabled as well.
<code>--dedup-min-qual</code>	Specifies the minimum Phred quality score for a base to be included in the quality score calculation used for choosing among duplicate reads.

Small Variant Calling

The DRAGEN Small Variant Caller is a high-speed haplotype caller implemented with a hybrid of hardware and software. The caller performs localized *de novo* assembly in regions of interest to generate candidate haplotypes, and then performs read likelihood calculations using a hidden Markov model (HMM).

Variant calling is disabled by default. To enable variant calling, set the `--enable-variant-caller` option to true. The VCF header is annotated with `##source=DRAGEN_SNV` to indicate the file is generated by the DRAGEN SNV pipeline.

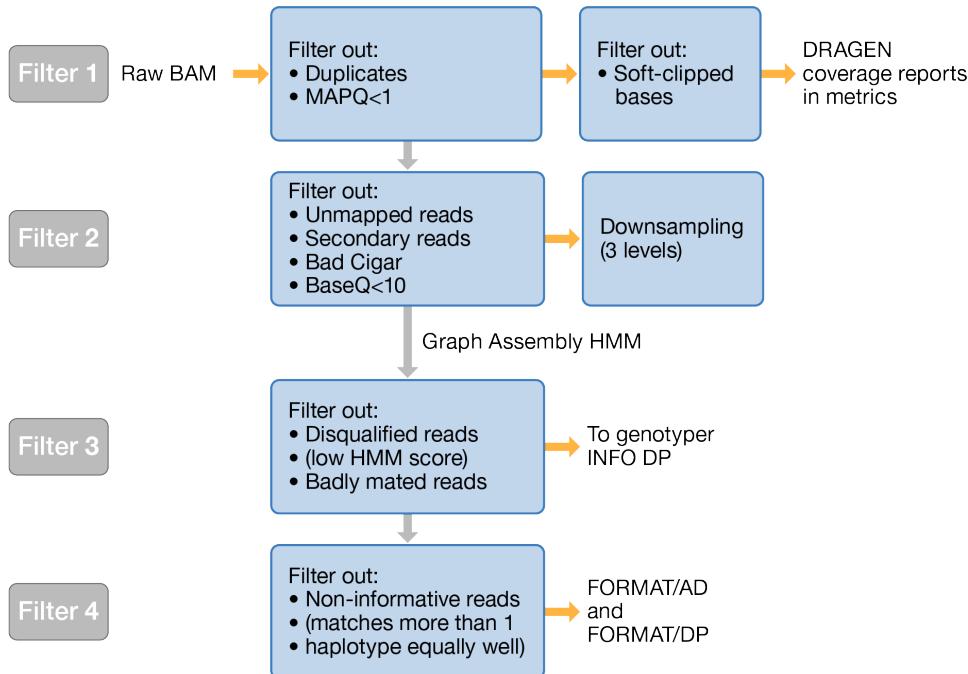
The Variant Caller Algorithm

The DRAGEN Small Variant Caller performs the following steps:

- Active Region Identification—Identifies areas where multiple reads disagree with the reference are identified, and selects windows around them (active regions) for processing.
- Localized Haplotype Assembly—For each active region, assembles all overlapping reads in each active region into a de Bruijn graph (DBG). A DBG is a directed graph based on overlapping Kmers (length K subsequences) in each read or multiple reads. When all reads are identical, the DBG is linear. Where there are differences, the graph forms bubbles of multiple paths diverging and rejoining. If the local sequence is too repetitive and K is too small, cycles can form, which invalidate the graph. Values of K=10 and 25 are tried by default. If those values produce an invalid graph, then additional values of K=35, 45, 55, 65 are tried until a cycle-free graph is obtained. From this cycle-free DBG, every possible path is extracted to produce a complete list of candidate haplotypes, ie, hypotheses for what the true DNA sequence may be on at least one strand.
- Localized Haplotype Assembly—Assembles all overlapping reads in each active region into a de Bruijn graph (DBG). A DBG is a directed graph based on overlapping K-mers (length K subsequences) in each read or multiple reads. When all reads are identical, the DBG is linear. Where there are differences, the graph forms bubbles of multiple paths that diverge and rejoin. If the local sequence is too repetitive and K is too small , cycles can form, which invalidate the graph. DRAGEN uses K=10 and 25 as the default values. If those values produce an invalid graph, then additional values of K=35, 45, 55, 65 are tried until a cycle-free graph is obtained. From this cycle-free DBG, DRAGEN extracts every possible path to produce a complete list of candidate haplotypes, ie, hypotheses for what the true DNA sequence might be on at least one strand.
- Haplotype Alignment—Uses the Smith-Waterman algorithm to align each extracted haplotype to the reference genome. The alignments determine what variations from the reference are present.
- Read Likelihood Calculation—Tests each read against each haplotype to estimate a probability of observing the read assuming the haplotype was the true original DNA sampled. This calculation is performed by evaluating a pair hidden Markov model (HMM), which accounts for the various possible ways the haplotype might have been modified by PCR or sequencing errors into the read observed. The HMM evaluation uses a dynamic programming method to calculate the total probability of any series of Markov state transitions arriving at the observed read.
- Genotyping—Forms the possible diploid combinations of variant events from the candidate haplotypes and, for each combination, calculates the conditional probability of observing the entire read pileup. Calculations use the constituent probabilities of observing each read, given each haplotype from the pair HMM evaluation. These calculations feed into the Bayesian formula to calculate the likelihood that each genotype is the genotype of the sample being analyzed, given the entire read pileup observed. Genotypes with maximum likelihood are reported.

Filter BAM Input

Information in the IGV BAM pileup differs from the INFO/DP and FORMAT/DP in the VCF/gVCF as a result of filtering steps applied throughout variant calling. There are four filters used to exclude reads from the genotyping calculations. The following figure summarizes the four filters.



- Filter 1 filters out the following reads from the IGV BAM input:
 - Duplicate reads.
 - Reads with MAPQ=0.
 - Soft-clipped bases. DRAGEN filters out soft-clipped bases only when calculating coverage reports.
 - [Somatic] Reads with MAPQ < vc-min-tumor-read-qual, where vc-min-tumor-read-qual > 1.
- Filter 2 trims bases with BQ < 10 and filters out the following reads:
 - Unmapped reads.
 - Secondary reads.
 - Reads with bad cigars.
- Filter 3 occurs after downsampling and HMM. Filter 3 filters out the following reads:
 - Reads that are badly mated. A badly mated read is a read where the pair is mapped to two different reference contigs.
 - Disqualified reads. Reads are disqualified if their HMM score is below a threshold.

- Filter 4 occurs after the genotyper runs. The genotyper adds annotation information to the FORMAT field. Filter 4 filters out reads that are not informative. For example, if the HMM scores of the read against two different haplotypes are almost equal, the read is filtered out because it does not provide enough information to distinguish which of the two haplotypes are more likely. INFO/DP includes both informative and non-informative reads. FORMAT/AD and FORMAT/DP include only informative reads.

Variant Caller Options

The following options control the variant caller stage.

Option	Description
--enable-variant-caller	Set <code>--enable-variant-caller</code> to <code>true</code> to enable the variant caller stage for the DRAGEN pipeline.
--vc-target-bed	[Optional] Restricts processing of the small variant caller, target BED related coverage, and callability metrics to regions specified in a BED file. The BED file is a text file containing at least three tab-delimited columns. The first three columns are chromosome, start position, and end position. The positions are zero-based. For example: <pre># header information chr11 0 246920 chr11 255660 255661</pre> If the reference span of the variant overlaps with any of the regions in the target BED, then the variant is output. If the reference span does not overlap, the variant is not output. For SNPs and Insertions, the reference span is 1 bp. For deletions, the reference span is the length of the deletion.
--vc-target-bed-padding	[Optional] Pads all target BED regions with the specified value. For example, if a BED region is 1:1000–2000 and the specified padding value is 100, the result is equivalent to using a BED region of 1:900–2100 and a padding value of 0. Any padding added to <code>--vc-target-bed-padding</code> is used by the small variant caller and by the target bed coverage/callability reports. The default padding is 0.
--vc-target-coverage	Specifies the target coverage for downsampling. The default value is 500 for germline mode and 50 for somatic mode.

Option	Description
--vc-enable-gatk-acceleration	Set to <code>true</code> to run the in GATK mode. Enabling GATK mode using this option is concordant with GATK 3.7 in germline mode and GATK 4.0 in somatic mode.
--vc-remove-all-soft-clips	Set to <code>true</code> to ignore soft-clipped bases during the haplotype assembly step.
--vc-decoy-contigs	Specifies a comma-separated list of contigs to skip during variant calling. This option can be set in the configuration file.
--vc-enable-decoy-contigs	Set to true to enable variant calls on the decoy contigs. The default value is false.
--vc-enable-phasing	Enable variants to be phased when possible. The default value is true.
--vc-combine-phased-variants-distance	Set the maximum distance between phased variants to be combined. The default value is 0, which disables the option. To enable the combining of phased variants the recommended value of the distance is 15 base pairs. The valid range is 0 to 15.

Downsampling Options for Small Variant Calling

You can use the following options for downsampling reads in the small variant calling pipeline.

Option	Description
--vc-target-coverage	Specifies the maximum number of reads with a start position overlapping any given position.
--vc-max-reads-per-active-region	Specifies the maximum number of reads covering a given active region.
--vc-max-reads-per-raw-region	Specifies the maximum number of reads covering a given raw region.
--vc-min-reads-per-start-pos	Specifies the minimum number of reads with a start position overlapping any given position.
--high-coverage-support-mode	Applies the high coverage mode down-sample options if set to true. Enabling this option is recommended for targeted panels with coverage over 1000x, but will slow down run time.

For mitochondrial small variant calling, the downsampling options can be set separately because the mitochondrial contig contains a higher depth than the rest of the contigs in a WGS data set. The following are the downsampling options for the mitochondrial contig.

- --vc-target-coverage-mito
- --vc-max-reads-per-active-region-mito

- `--vc-max-reads-per-raw-region-mito`

The target coverage and max/min reads in raw/active region options are not directly related and could be triggered independently.

The target coverage option runs first and is meant to limit the number of reads that share the same start position at any given position. It is not a limit on the total coverage at a given position.

The following are the default downsampling values for each small variant calling mode.

Mode	Downsampling Option	Default Value
Germline	<code>--vc-target-coverage</code>	500
Germline	<code>--vc-max-reads-per-active-region</code>	10000
Germline	<code>--vc-max-reads-per-raw-region</code>	30000
Somatic	<code>--vc-target-coverage</code>	50
Somatic	<code>--vc-max-reads-per-active-region</code>	10000
Somatic	<code>--vc-max-reads-per-raw-region</code>	30000
High Coverage	<code>--vc-target-coverage</code>	100000
High Coverage	<code>--vc-max-reads-per-active-region</code>	200000
High Coverage	<code>--vc-max-reads-per-raw-region</code>	200000
Mitochondrial	<code>--vc-target-coverage-mito</code>	40000
Mitochondrial	<code>--vc-max-reads-per-active-region-mito</code>	200000
Mitochondrial	<code>--vc-max-reads-per-raw-region-mito</code>	200000

The following example shows that the DP reported in a variant record can exceed the `--vc-target-coverage` default value of 500 in germline mode:

For example, assume the default value of `--vc-target-coverage` is 500. If there are 400 reads starting at position 1, another 400 starting at position 2, and another 400 starting at position 3, the target coverage option is not triggered (because $400 < 500$). If there is a variant at position 4, reported depth of the variant could be as high as 1200. This example shows that the DP reported in a variant record can exceed the `--vc-target-coverage` value.

After the target coverage step, the maximum number of reads that share the same position is 500 (if `--vc-target-coverage` is set to 500).

The next downsampling step is to apply the `--vc-max-reads-per-raw-region` and `--vc-max-reads-per-active-region` limits. In this step, the maximum number of reads that share the same position can be further reduced from the 500 maximum value from the first step. These options are used to limit the total number of reads in an entire region using a leveling downsampling method.

The downsampling mechanism scans each start position from the start boundary of the region and discards one read from that position, then moves on to the next position, until the total number of reads falls below the threshold. It can potentially take several passes across the entire region for the total

number of reads in the entire region to fall below the threshold. After the threshold is met, the downsampling step is stopped regardless of which position was considered last in the region.

If the number of reads at any position with same start position is equal to or lower than the `--vc-min-reads-per-start-pos`, that position is skipped to make sure that there is always at least a minimum number of reads (set to `--vc-min-reads-per-start-pos`) at any start position.

When downsampling occurs, the choice of which reads to keep or remove is random. However, the random number generator is seeded to a default value to make sure that the generator produces the same set of values in each run. This ensures reproducible results, which means there is no run to run variation when using the same input data.

gVCF Output

A genomic VCF (gVCF) file contains information on variants and positions determined to be homozygous to the reference genome. For homozygous regions, the gVCF file includes statistics that indicate how well reads support the absence of variants or alternative alleles. The gVCF file includes an artificial <NON_REF> allele. Reads that do not support the reference or any variants are assigned the <NON_REF> allele. DRAGEN uses these reads to determine if the position can be called as a homozygous reference, as opposed to remaining uncalled. The resulting score represents the Phred-scaled level of confidence in a homozygous reference call. In germline mode, the score is FORMAT/GQ and in somatic mode the score is FORMAT/SQ.

The following are the available gVCF output options.

Option	Description
<code>--vc-emit-ref-confidence</code>	To enable gVCF output, set to GVCF. By default, contiguous runs of homozygous reference calls with similar scores are collapsed into blocks (hom-ref blocks). Hom-ref blocks save disk space and processing time of downstream analysis tools. DRAGEN recommends using the default mode. To produce unbanded output, set <code>--vc-emit-ref-confidence</code> to BP_RESOLUTION.
<code>--vc-enable-vcf-output</code>	To enable VCF file output during a gVCF run, set to true. The default value is false.
<code>--vc-gvcf-bands</code>	If using the default <code>--vc-emit-ref-confidence gvcf</code> (banded mode), DRAGEN collapses gVCF records with a similar GQ or SQ score. By default, the cutoffs are 1 10 20 30 40 60 80 for germline and 1 3 10 20 50 80 for somatic. For example, to define the bands [0, 10), [10, 50), and ≥ 50 use <code>--vc-gvcf-bands 10 50</code> .

Option	Description
--vc-compact-gvcf	This option, when used for germline in conjunction with --vc-emit-ref-confidence gvcf, produces a much smaller gVCF output file than the default. It can be used when the gVCF is destined for ingestion into gVCF Genotyper, offering further savings on disk space and gVCF Genotyper runtime compared to the default. The option implies --vc-gvcf-bands 0 1 10 20 30 and omits certain metrics that are not used by gVCF Genotyper. i Files generated using this option will be rejected by the Pedigree Caller.

Not all entries in the gVCF are contiguous. The file might contain gaps that are not covered by either a variant line or a hom-ref block. The gaps correspond to regions that are not callable. A region is not callable if there is not at least one read mapped to the region with a MAPQ score above zero.

In germline mode, the thresholds for calling are lower for gVCFs than for VCFs. The gVCF output could show a different number of variants than a VCF run for the same sample. There is likely a different number of biallelic and multiallelic calls because gVCF mode includes all possible alleles at a locus, rather than only the two most likely alleles. This means that a biallelic call in the VCF can be output as a multiallelic call in the gVCF. The genotype in the gVCF still points to the two most likely alleles, so the variant call remains the same.

The following are example gVCF records that include a hom-ref block call and a variant call.

```
1 39224 . C <NON_REF> . PASS END=39260
GT:AD:DP:GQ:MIN_DP:PL:SPL:ICNT
0/0:2,0:2:3:1:0,3,37:0,3,37:3,0
1 39261 . T C,<NON_REF> 15.59 PASS
DP=3;MQ=12.73;MQRankSum=0.736;ReadPosRankSum=0.736;FractionInformativeReads=1.0
00
GT:AD:AF:DP:F1R2:F2R1:GQ:PL:SPL:ICNT:GP:PRI:SB:MB
0/1:1,2,0:0.667,0.000:3:1,0,0:0,2,0:5:49,0,1,69,7,75:66,0,8:1,0:1.5592e+01,1.59
15e+
00,5.5412e+00,7.0100e+01,4.3330e+01,8.0068e+01:0.00,34.77,37.77,34.77,69.54,37.
77:0,1,0,2:0,1,2,0
```

In a single sample gVCF, FORMAT/DP reported at a HomRef position is the median DP in the band and, AD is the corresponding value. The sum of AD will be DP even in a homref band. The minimum is computed and printed as MIN_DP for the band.

QUAL, QD, and GQ

In single sample VCF and gVCF, the QUAL follows the definition of the VCF specification. For more information on the VCF specification, see the most current VCF documentation available on samtools/hts-specs GitHub repository.

- QUAL is the Phred-scaled probability that the site has no variant and is computed as:

```
QUAL = -10*log10 (posterior genotype probability of a homozygous-reference
genotype (GT=0/0))
```

That is, $\text{QUAL} = \text{GP}(\text{GT}=0/0)$, where GP is the posterior genotype probability in Phred scale.

$\text{QUAL} = 20$ means there is 99% probability that there is a variant at the site. The GP values are also given in Phred-scaled in the VCF file.

- GQ is the Phred-scaled probability that the call is incorrect.

$\text{GQ} = -10 \log_{10}(p)$, where p is the probability that the call is incorrect.

$\text{GQ} = -10 \log_{10}(\sum(10^{-\text{GP}(i)/10}))$ where the sum is taken over the GT that did not win.

GQ of 3 indicates a 50 percent chance that the call is incorrect, and GQ of 20 indicates a 1 percent chance that the call is incorrect.

- QD is the QUAL normalized by the read depth, DP.

The formulation is summarized in the following table.

Metric	QUAL	GQ	QD
Description	Probability that the site has no variant	Probability that the call is incorrect	Qual normalized by Depth
Formulation	$\text{QUAL} = \text{GP}(\text{GT}=0/0)$	$\text{GQ} = -10 \log_{10}(p)$	QUAL/DP
Scale	Unsigned Phred	Unsigned Phred	Unsigned Phred
Numerical example	QUAL=20: 1 % chance that there is no variant at the site QUAL=50: 1 in 1e5 chance that there is no variant at the site	GQ=3, 50% chance that the call is incorrect GQ=20, 1% chance that the call is incorrect	

Phasing and Phased Variants

DRAGEN supports output of phased variant records in the germline VCF and gVCF file. When two or more variants are phased together, the phasing information is encoded in a sample-level annotation, FORMAT/PS. FORMAT/PS identifies which set the phased variant is in. The value in the field is an integer representing the position of the first phased variant in the set. All records in the same contig with matching PS values belong to the same set.

```
##FORMAT=<ID=PS,Number=1,Type=Integer,Description="Physical phasing ID information, where each unique ID within a given sample (but not across samples) connects records within a phasing group">
```

The following is an example of a DRAGEN single sample gVCF, where two SNPs are phased together.

```
chr1 1947645 . C T,<NON_REF> 48.44 PASS
DP=35;MQ=250.00;MQRankSum=4.983;ReadPosRankSum=3.217;FractionIninformativeReads=1
.000;R2_5P_bias=0.000 GT:AD:AF:DP:F1R2:F2R1:GQ:PL:SPL:ICNT:GP:PRI:SB:MB:PS
0|1:20,15,0:0.429:35:9,7,0:11,8,0:47:83,0,50,572,758,622:255,0,255:19,0:4.844e+
01,8.387e-
05,5.300e+01,4.500e+02,4.500e+02,4.500e+02:0.00,34.77,37.77,34.77,69.54,37.77:1
1,9,10,5:12,8,8,7:1947645

chr1 1947648 . G A,<NON_REF> 50.00 PASS
DP=36;MQ=250.00;MQRankSum=5.078;ReadPosRankSum=2.563;FractionIninformativeReads=1
.000;R2_5P_bias=0.000 GT:AD:AF:DP:F1R2:F2R1:GQ:PL:SPL:ICNT:GP:PRI:SB:MB:PS
1|0:16,20,0:0.556:36:8,9,0:8,11,0:48:85,0,49,734,613,698:255,0,255:16,0:5.000e+
01,7.067e-
05,5.204e+01,4.500e+02,4.500e+02,4.500e+02:0.00,34.77,37.77,34.77,69.54,37.77:1
0,6,11,9:8,8,12,8:1947645
```

During the genotyping step, all haplotypes and all variants are considered over an active region. For each pair of variants, if both variants occur on all of the same haplotypes or if either is a homozygous variant, then they are phased together. If the variants only occur on different haplotypes, then they are phased opposite to each other. If any heterozygous variants are present on some of the same haplotypes but not others, phasing is aborted and no phasing information is output for the active region.

Combine Phased Variants

Phased variant records that belong to the same phasing set can be combined into a single VCF record. For example, assuming reference at position chr2 115035 is A, the following two phased variants are combined.

```
chr2 115034 . G C GT:PS 0|1:115034
chr2 115036 . C T GT:PS 0|1:115034
```

The phased variants are combined as follows.

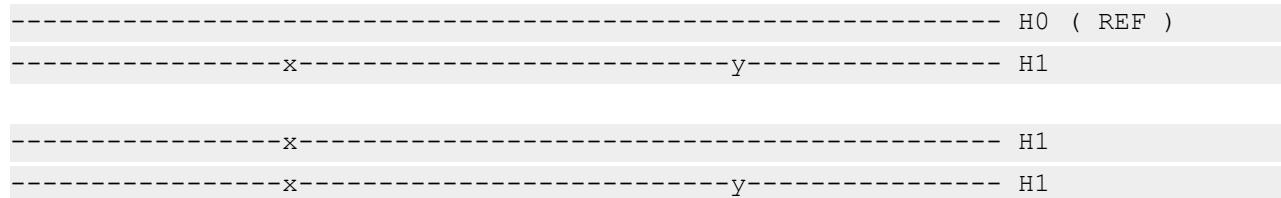
```
chr2 115034 . GAC CAT GT:PS 0|1:115034
```

The command line option `--vc-combine-phased-variants-distance` specifies the maximum distance over which phased variants will be combined. The default value 0 disables the feature. When enabled, the option combines all phased variants in the phasing set that are within the provided distance value.

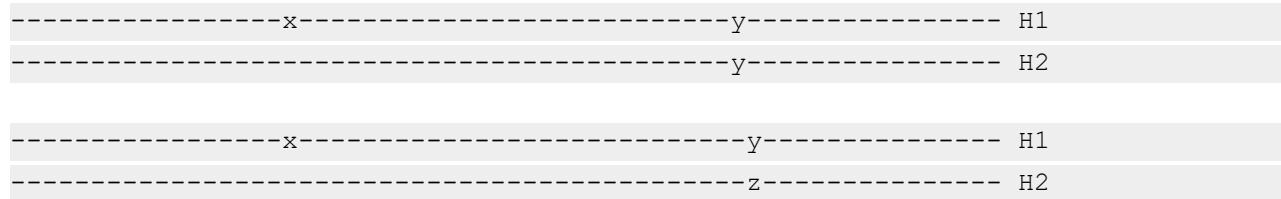
DRAGEN supports phasing of the genotypes listed in the below table. Only the first row in the table is relevant to somatic, because the somatic pipeline only emits 0/1 and 0|1 genotypes. MNV calls can still be phased with other variant calls that fell outside the phased variants distance.

GT variant 1	GT variant 2	GT MNV	Relevant Pipeline	Supported in DRAGEN
0 1	0 1	0/1	Germline and Somatic	Yes in 4.0
0/1	1/1	1/2	Germline	No
0/1	1/2	1/2	Germline	No
0/1	1/1	1/1	Germline	Yes in 4.2

Examples of diploid haplotypes where phasing is supported:



Examples of diploid haplotypes where phasing is not supported:



By default, DRAGEN will output component SNVs/INDELs of MNV calls only if the VAF of the component call is greater than that of the MNV by more than 0.1. The VAF difference threshold for outputting component calls along with MNV calls is controlled by the `--vc-combine-phased-variants-max-vaf-delta` option. DRAGEN also offers the `--vc-mnv-emit-component-calls` option to output all component SNVs/INDELs of MNVs, regardless of VAF difference, when enabled. These two options are mutually exclusive and are only available for use in the somatic pipeline.

Ploidy Support

The small variant caller currently only supports either ploidy 1 or 2 on all contigs within the reference except for the mitochondrial contig, which uses a continuous allele frequency approach (see [Mitochondrial Calling on page 122](#)). The selection of ploidy 1 or 2 for all other contigs is determined as follows.

- If `--sample-sex` is not specified on the command line, the Ploidy Estimator determines the sex. If the Ploidy Estimator cannot determine the sex karyotype or detects sex chromosome aneuploidy, all contigs are processed with ploidy 2.

- If `--sample-sex` is specified on the command line, contigs are processed as follows.
 - For female samples, DRAGEN processes all contigs with ploidy 2 and marks variant calls on chrY with a filter PloidyConflict.
 - For male samples, DRAGEN processes all contigs with ploidy 2, except for the sex chromosomes. DRAGEN processes chrX with ploidy 1, except in the PAR regions, where it is processed with ploidy 2. chrY is processed with ploidy 1 throughout.

For male samples in germline calling mode, DRAGEN calls potential mosaic variants in non-PAR regions of sex chromosomes. A variant is called as mosaic when the allele frequency (FORMAT/AF) is below 85% or if multiple alt alleles are called, suggesting incompatibility with the haploid assumption. The GT field for bi-allelic mosaic variants is "0/1", denoting a mixture of reference and alt alleles, as opposed to the regular GT of "1" for haploid variants. The GT field for multi-allelic mosaic variants is "1/2" in VCF. You can disable the calling of mosaic variants by setting `--vc-enable-sex-chr-diploid` to false.

An example germline VCF record of a mosaic variant in a haploid region: chrX 18622368 . C T 48.84 PASS

```
AC=1;AF=0.500;AN=2;DP=22;FS=4.154;MQ=248.02;MQRankSum=3.272;QD=2.27;ReadPosRankSum=2.671;SOR=1.546;FractionInformativeReads=1.000;MOSAIC
GT:AD:AF:DP:F1R2:F2R1:GQ:PL:GP:PRI:SB:MB
0/1:9,13:0.5909:22:1,8:5:48:84,0,51:4.8837e+01,7.4031e-05,5.4007e+01:0.00,34.77,37.77:5,4,4,9:3,6,5,8
```

DRAGEN detects sex chromosomes by the naming convention, either X/Y or chrX/chrY. No other naming convention is supported.

<code>--sample-sex</code>	Ploidy Estimation	Sample Sex in Small Variant Caller
male	Not relevant	Male
female	Not relevant	Female
None	Not relevant	None
auto (default)	XY	Male
auto (default)	XX	Female
auto (default)	Everything else	None

Overlapping Mates in Small Variant Calling

Instead of treating overlapping mates as independent evidence for a given event, DRAGEN handles overlapping mates in both the germline and somatic pipelines as follows.

- When the two overlapping mates agree with each other on the allele with the highest HMM score, the genotyper uses the mate with the greatest difference between the highest and the second highest HMM score. The HMM score of the other mate becomes zero.

- When the two overlapping mates disagree, the genotyper sums the HMM score between the two mates, assigns the combined score to the mate that agrees with the combined result, and changes the HMM score of the other mate to zero.
- The base qualities of overlapping mates are no longer adjusted.

Mitochondrial Calling

Typically, there are approximately 100 mitochondria in each mammalian cell. Each mitochondrion harbors 2–10 copies of mitochondrial DNA (mtDNA). For example, if 20% of the chrM copies have a variant, then the allele frequency (AF) is 20%. This is also referred to as continuous allele frequency. The expectation is that the AF of variants on chrM is anywhere between 0% and 100%.

DRAGEN processes chrM through a continuous AF pipeline. In this case, a single ALT allele is considered and the AF is estimated. The estimated AF can be anywhere between 0% and 100%. The default variant AF thresholds are applied to mitochondrial variant calling.

- `--vc-enable-af-filter-mito`
Whether to enable the allele frequency for mitochondrial variant calling. The default is true.
- `--vc-af-call-threshold-mito`
Set the threshold for emitting calls in the VCF. The default is 0.01.
- `--vc-af-filter-threshold-mito`
Set the threshold to mark emitted vcf call as filtered. The default is 0.02.

QUAL and GQ are not output in the chrM variant records. Instead, the confidence score is FORMAT/SQ, which gives the Phred-scaled confidence that a variant is present at a given locus. A call is made if `FORMAT/SQ > vc-sq-call-threshold` (default = 3.0).

```
##FORMAT=<ID=SQ,Number=A>Type=Float,Description="Somatic quality">
```

You can apply the following filters to mitochondrial variant calls.

- `--vc-sq-call-threshold`
Set the SQ threshold for emitting calls in the VCF. The default is 0.1.
- `--vc-sq-filter-threshold`
Set the SQ threshold to mark emitted VCF calls as filtered. The default is 3.0.
- `--vc-enable-triallelic-filter`
Enable the multiallelic filter. The default value is false.
- If `FORMAT/SQ < vc-sq-call-threshold`, the variant is not output in the VCF.
- If `FORMAT/SQ > vc-sq-call-threshold` but `FORMAT/SQ < vc-sq-filter-threshold` the variant is output in the VCF but `FILTER=weak_evidence`.
- If `FORMAT/SQ > vc-sq-call-threshold`, `FORMAT/SQ > vc-sq-filter-threshold`, and no other filters are triggered, the variant is output in the VCF and `FILTER=PASS`.

The following are example VCF records on the chrM. The examples show one call with very high AF and another with low AF. In both cases FORMAT/SQ > vc-sq-call-threshold. FORMAT/SQ is also > vc-sq-filter-threshold, so the FILTER annotation is PASS.

```
chrM 513 . GCA G . PASS DP=4937;MQ=235.28;FractionInformativeReads=0.883
GT:SQ:AD:AF:F1R2:F2R1:DP:SB:MB
1/1:95.46:33,4327:0.992:7,1081:26,3246:4360:31,2,2371,1956:10,23,2811,1516
chrM 7028 . C T . PASS DP=8868;MQ=60.19;FractionInformativeReads=0.993
GT:SQ:AD:AF:F1R2:F2R1:DP:SB:MB
0/1:21.48:8622,181:0.021:4190,82:4432,99:8803:4344,4278,94,87:5032,3590,101,80
```

FORMAT/GT

For homref calls (eg in NON_REF regions of gVCF output) the FORMAT/GT is hard-coded to 0/0. The FORMAT/AF yields an estimate on the variant allele frequency, which ranges anywhere within [0,1]. For variant calls with FORMAT/AF < 95%, the FORMAT/GT is set to 0/1. For variants with very high allele frequencies (FORMAT/AF \geq 95%), the FORMAT/GT is set to 1/1.

The following is an example of a variant record on chrM in a trio joint VCF. The variant was detected in the second sample with a confidence score that passed the filter threshold. In the first and third samples GT=0/0, which indicates a tentative hom-ref call (ie, that position for the sample is in a NON_REF region over which no variant was detected with sufficient confidence), but the weak_evidence filter tag indicates that this call is made with low confidence.

```
chrM 2623 . A G . PASS DP=18772;MQ=111.77 GT:AD:AF:DP:FT:SQ:F1R2:F2R1
0/0:6841,7:0.001:4334:weak_evidence:0:...
0/1:6736,2053:0.234:8789:PASS:21.32:3394,1060:3342,993
0/0:6086,9:0.001:5613:weak_evidence:0:...
```

Joint Detection of Overlapping Variants

When variants at multiple loci in a single active region are detected jointly, genotyping can benefit. DRAGEN combines loci into a joint detection region if the following conditions are met:

- Loci have alleles that overlap each other.
- Loci are in the STR region or less than 10 bases apart of the STR region.
- Loci are less than 10 bases apart of each other.

Joint detection generates a haplotype list where all possible combinations of the alleles in the joint detection regions are represented. This calculation leads to a larger number of haplotypes. During genotyping, joint detection calculates the likelihoods that each haplotype pair is the truth, given the observed read pileup. Genotype likelihoods are calculated as the sum of the likelihoods of haplotype pairs that support the alleles in the genotypes. Genotypes with maximum likelihood are reported.

Joint detection is enabled by default. To disable joint detection, set --vc-enable-joint-detection to false.

ROH Caller

Regions of homozygosity (ROH) are detected as part of the small variant caller. The caller detects and outputs the runs of homozygosity from whole genome calls on autosomal human chromosomes. Sex chromosomes are ignored unless the sample sex karyotype is XX, as specified on the command line or determined by the Ploidy Estimator. ROH output allows downstream tools to screen for and predict consanguinity between the parents of the proband subject.

A region is defined as consecutive variant calls on the chromosome with no large gap in between these variants. In other words, regions are broken by chromosome or by large gaps with no SNV calls. The gap size is set to 3 Mbases.

ROH Algorithm

The ROH algorithm runs on the small variant calls. The algorithm excludes variants with multiallelic sites, indels, complex variants, non-PASS filtered calls, and homozygous reference sites. The variant calls are then filtered further using a block list BED, and finally depth filtering is applied after the block list filter. The default value for the fraction of filtered calls is 0.2, which filters the calls with the highest 10% and lowest 10% in DP values. The algorithm then uses the resulting calls to find regions.

The ROH algorithm first finds seed regions that contain at least 50 consecutive homozygous SNV calls with no heterozygous SNV or gaps of 500,000 bases between the variants. The regions can be extended using a scoring system that functions as follows.

- Score increases with every additional homozygous variant (0.025) and decreases with a large penalty (1–0.025) for every heterozygous SNV. This provides some tolerance of presence of heterozygous SNV in the region.
- Each region expands on both ends until the regions reach the end of a chromosome, a gap of 500,000 bases between SNVs occurs, or the score becomes too low (0).

Overlapping regions are merged into a single region. Regions can be merged across gaps of 500,000 bases between SNVs if a single region would have been called from the beginning of the first region to the end of the second region without the gap. There is no maximum size for regions, but regions always end at chromosome boundaries.

ROH Options

- `--vc-enable-roh`—Set to true to enable the ROH caller. The ROH caller is enabled by default for human autosomes only. Set to false to disable.
- `--vc-roh-blacklist-bed`—If provided, the ROH caller ignores variants that are contained in any region in the exclude-BED file. DRAGEN distributes exclude-BED files for all popular human genomes and automatically selects a file to match the genome in use. Unless this option is used explicitly, select a file.

ROH Output

The ROH caller produces an ROH output file named `<output-file-prefix>.roh.bed` in which each row represents one region of homozygosity. The BED file contains the following columns:

Chromosome	Start	End	Score	#Homozygous	#Heterozygous
------------	-------	-----	-------	-------------	---------------

- Score is a function of the number of homozygous and heterozygous variants, where each homozygous variant increases the score by 0.025, and each heterozygous variant reduces the score by 0.975.
- Start and end positions are a 0-based, half-open interval.
- #Homozygous is number of homozygous variants in the region.
- #Heterozygous is number of heterozygous variants in the region.

The caller also produces a metrics file named `<output-file-prefix>.roh_metrics.csv` that lists the number of large ROH and percentage of SNPs in large ROH (> 3 MB).

B-Allele Frequency Output

B-Allele frequency (BAF) output is enabled by default in germline and somatic VCF and gVCF runs.

The BAF value is equal to either AF or (1–AF), where

- $AF = (\text{alt_count} / (\text{ref_count} + \text{alt_count}))$
- $BAF = 1 - AF$, only when ref base < alt base, order of priority for bases is A < T < G < C < N

For each small variant VCF entry with exactly one SNP alternate allele, the output contains a corresponding entry in the BAF output file.

- <NON_REF> lines are excluded
 - ForceGT variants (as marked by the "FGT" tag in the INFO field) are not included in the output, unless the variant also contains the "NML" tag in the INFO field.
 - Variants where the ref_count and alt_count are both zero are not included in the output.

BAF Options

`--vc-enable-baf`

Enable or disable B-allele frequency output. Enabled by default.

BAF Output

The BF generates are BigWig-compressed files, named `<output-file-prefix>.baf.bw` and `<output-file-prefix>.hard-filtered.baf.bw`. The hard-filtered file only contains entries for variants that pass the filters defined in the VCF (ie, PASS entries).

Each entry contains the following information:

Chromosome	Start	End	BAF
------------	-------	-----	-----

Where:

- Chromosome is a string matching a reference contig.
- Start and end values are zero-based, half open intervals.
- BAF is a floating point value.

Somatic Mode

The DRAGEN Somatic Pipeline allows ultrarapid analysis of Next-Generation Sequencing (NGS) data to identify cancer-associated mutations in somatic chromosomes. DRAGEN calls SNVs and indels from both matched tumor-normal pairs and tumor-only samples using a probability model that considers the possibility of somatic variants, germline variants, and various systematic noise artifacts. When considering somatic variants, DRAGEN does not make any ploidy assumptions, which enables detection of low-frequency alleles. For loci with coverage up to 100x in the tumor sample, DRAGEN has a limit of detection at variant allele frequencies of 5%. The limit scales with increasing depth on a per-locus basis and halves every time the coverage doubles beyond 100x.

For the tumor-normal pipeline, both samples are analyzed jointly. DRAGEN assumes that germline variants and systematic noise artifacts would be shared by both samples, whereas somatic variants would be present only in the tumor sample. Only somatic variants are reported. To detect systematic noise artifacts, DRAGEN recommends that the coverage in the normal sample be at least half of the coverage in the tumor sample. The tumor-only pipeline produces output containing both germline and somatic variants that can be further analyzed to identify tumor mutations. The caller attempts to distinguish between them based on allele frequency and an assumption that germline variants occur in the absence of copy number variation (CNV), labeling the variants deemed to be somatic with the `SOMATIC` tag in the info field. We caution that this labeling may be misleading in the presence of copy number variation, and that filtering out common germline variants as reported in databases may be a more reliable way to remove germline variants if that is desired. However, the labeling should be useful in the absence of copy number variation, and in particular for the purpose of constructing systematic noise bed files from normal samples, see Systematic Noise BED File Generation.

After multiple filtering steps, the output is generated as a VCF file. Variants that fail the filtering steps are kept in the output VCF. The variants include a `FILTER` annotation that indicates which filtering steps have failed.

DRAGEN uses a Bayesian approach to compute the posterior probability that a somatic variant is present and reports this as a Phred-scaled quantity, "somatic quality" (SQ). This is done by computing likelihoods for several hypotheses and noise processes, taking into account many factors such as: the numbers of alt-supporting and ref-supporting reads in the tumor and normal samples (and hence the alt allele frequencies in both samples); mapping qualities and how these are distributed across the reads in the tumor and normal pile ups; base call qualities; forward vs reverse strand support; sample-wide estimates of insertion and deletion error probabilities as functions of repeat period, repeat length, and indel length; sample-wide estimates of nucleotide error biases; whether there are nearby co-phased events; and whether the positions in question are known somatic hotspots or associated with sequence-specific error patterns. You can use SQ as the primary metric to describe the confidence with

which the caller made a somatic call. SQ is reported as a format field for the tumor sample (exception: for homozygous reference calls in gVCF mode it is instead a likelihood ratio, analogous to hom-ref GQ as described in the germline section). Variants with SQ score below the SQ filter threshold are filtered out using the `weak_evidence` tag. To trade off sensitivity against specificity, adjust the SQ filter threshold. Lower thresholds produce a more sensitive caller and higher thresholds produce a more conservative caller. If performing tumor-normal analysis, the SQ field for the normal sample contains the Phred-scaled posterior probability that a putative call is a germline variant.

Somatic Mode Options

Somatic mode has the following command line options:

Option	Description
<code>--tumor-fastq1</code>	Inputs a pair of FASTQ files into the mapper aligner and somatic variant caller. You can use these options with OTHER FASTQ options to run in tumor-normal mode.
<code>--tumor-fastq2</code>	<p>For example:</p> <pre>dragen -f -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \ --tumor-fastq1 <TUMOR_FASTQ1> \ --tumor-fastq2 <TUMOR_FASTQ2> \ --RGID-tumor <RG0-tumor> --RGSM-tumor <SM0-tumor> \ -1 <NORMAL_FASTQ1> \ -2 <NORMAL_FASTQ2> \ --RGID <RG0> -RGSM <SM0> \ --enable-variant-caller true \ --output-directory /staging/examples/ \ --output-file-prefix SRA056922_30x_e10_50M</pre>
<code>--tumor-fastq-list</code>	Inputs a list of FASTQ files into the mapper aligner and somatic variant caller. You can use these options with other FASTQ options to run in tumor-normal mode. For example:
	<pre>dragen -f \ -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149 \ --tumor-fastq-list <TUMOR_FASTQ_LIST> \ --fastq-list <NORMAL_FASTQ_LIST> \ --enable-variant-caller true \ --output-directory /staging/examples/ \ --output-file-prefix SRA056922_30x_e10_50M</pre>

Option	Description
--tumor-bam-input --tumor-cram-input	Inputs a mapped BAM or CRAM file into the somatic variant caller. You can use these options with other BAM/CRAM options to run in tumor-normal mode.
--vc-target-vaf	The --vc-target-vaf is used to select the variant allele frequencies of interest. The variant caller will aim to detect variants with allele frequencies larger than this setting. We recommend adding a small safety factor, e.g. to ensure variants in the ballpark of 1% are detected, the minimum vc-target-vaf can be specified as 0.009 (0.9%). This setting will not apply a hard threshold, and it is possible to detect variants with allele frequencies lower than the selected threshold. On high coverage and clean datasets, a lower target-vaf may help increase sensitivity. On noisy samples (like FFPE) a higher target-vaf (like 0.03) maybe help reduce false positives. Using a low target-vaf may also increase runtime. Set the vc-target-vaf to 0 to disable this feature. When this feature is disabled the variant caller will require at least 2 supporting reads to discover a candidate variant. Valid range=[0, 1], default=0.01.
--vc-base-quality-threshold	The minimum BQ that will be used at any stage in the SNV caller.
--vc-min-tumor-read-quality	Specifies the minimum read quality (MAPQ) to be considered for variant calling. The default value is 3 for tumor-normal analysis or 20 for tumor-only analysis.
--vc-callability-tumor-threshold	Specifies the callability threshold for tumor samples. The somatic callable regions report includes all regions with tumor coverage above the tumor threshold. The default value is 15. For more information on the somatic callable regions report, see Somatic Callable Regions Report on page 349
--vc-callability-normal-threshold	Specifies the callability threshold for normal samples, if present. If applicable, the somatic callable regions report includes all regions with normal coverage above the normal threshold. The default value is 5. For more information on the somatic callable regions report, see Somatic Callable Regions Report on page 349 .

Option	Description
--vc-somatic-hotspots	Use <code>--vc-somatic-hotspots</code> to specify a hotspots input VCF. The variants in the hotspots VCF indicate positions where more somatic mutations are expected. For example, a hotspots input VCF can be derived from COSMIC. DRAGEN genotyping priors are boosted for all positions specified in the VCF, so you can call a variant at one of these sites with fewer supporting reads. If you do not provide a value for <code>vc-somatic-hotspots</code> , DRAGEN automatically selects a reference-specific hotspots VCF file from <code>/opt/edico/config/somatic_hotspots*</code> . If you provide a <code>vc-somatic-hotspots</code> VCF, the VCF always takes precedence over the default hotspots VCF. To disable the hotspots feature, use <code>vc-use-somatic-hotspots=false</code> . Neither the default VCF nor a provided VCF is considered.
--vc-hotspot-log10-prior-boost	i Input VCF records must be sorted in the same order as contigs in the selected reference.
--vc-hotspot-log10-prior-boost	If using <code>--vc-use-somatic-hotspots</code> , use <code>vc-hotspotlog10-prior-boost</code> to control the size of the adjustment. The default value is 4 (log10 scale) corresponding to an increase of 40 Phred.

Option	Description
--vc-enable-liquid-tumor-mode --vc-tin-contam-tolerance	In a tumor-normal analysis, DRAGEN accounts for tumor-in-normal (TiN) contamination by running liquid tumor mode. Liquid tumor mode is disabled by default. When liquid tumor mode is enabled, DRAGEN is able to call variants in the presence of TiN contamination up to a specified maximum tolerance level. --vc-enable-liquid-tumor-mode enables liquid tumor mode with a default maximum contamination TiN tolerance of 0.15. If using the default maximum contamination TiN tolerance, somatic variants are expected to be observed in the normal sample with allele frequencies up to 15% of the corresponding allele in the tumor sample. vc-tin-contam-tolerance enables liquid tumor mode and allows you to set the maximum contamination TiN tolerance. Liquid tumor mode is not equivalent to liquid biopsy. Liquid tumors in liquid tumor mode refer to hematological cancers, such as leukemia. For liquid tumors, it is not feasible to use blood as a normal control because the tumor is present in the blood. Skin or saliva is typically used as the normal sample. However, skin and saliva samples can still contain blood cells, so that the matched normal control sample contains some traces of the tumor sample and somatic variants are observed at low frequencies in the normal sample. If the contamination is not accounted for, it can severely impact sensitivity by suppressing true somatic variants.
	Liquid tumor mode typically uses a library that is WGS or WES with medium depth for example (100x T/ 40xN), and the lowest VAF detected for these types of depths is ~5%. Liquid biopsy typically uses a targeted gene panel (eg, 500 genes), with very high raw depth, and uses UMI indexing (collapsing down to a depth of >2000x) to enable sensitivity at VAF down to 0.1 % in some cases (the limit of detection will vary depending on coverage and data quality).
--vc-override-tumor-pcr-params-with-normal	If using different sequencing systems or different library preparation methods for tumor and normal samples, DRAGEN recommends setting this option to false. In tumor-normal mode, DRAGEN estimates a set of PCR error parameters separately for each of the tumor and normal samples. By default, DRAGEN ignores the tumor-sample parameters and uses normal-sample parameters for analysis of both samples. This default prevents overestimation of tumor-sample error rates that can occur if the somatic variant rate is high. For example, the error rates can occur in germline admixture data sets where the somatic variant rate is equal to the germline variant rate.

Option	Description								
--vc-enable-unequal-ntd-errors	These options control the Sample-specific NTD Error Bias Estimation on page 131 feature.								
--vc-enable-trimer-context									
--vc-snp-error-cal-bed									
--vc-af-lod	The probability calculation in the somatic caller assesses variant and noise hypotheses at fixed allele frequencies defined by a discrete grid (by default at coverages <200: 0, 0.05, 0.1, ... 1.0). This means that the calculation will assess variants with allele frequencies below 0.05 as if the true frequency is equal to 0.05; this strategy does not preclude such variants from being called but may result in lower scores compared to if the true frequency had been considered. At positions with higher coverage, DRAGEN adds extra grid points as in the table below to consider hypotheses involving lower allele frequencies and effectively achieve a lower limit of detection (LOD), with the lowest VAF halving every time the coverage doubles:								
<table border="1"> <thead> <tr> <th>Coverage</th><th>Lowest AF</th></tr> </thead> <tbody> <tr> <td>0-199</td><td>0.05</td></tr> <tr> <td>200-399</td><td>0.025</td></tr> <tr> <td>400-799</td><td>0.0125</td></tr> </tbody> </table>		Coverage	Lowest AF	0-199	0.05	200-399	0.025	400-799	0.0125
Coverage	Lowest AF								
0-199	0.05								
200-399	0.025								
400-799	0.0125								
<p>To make sure that variant hypotheses at a specific LOD are always considered even at positions with low coverage. You can use the <code>vc-af-lod</code> option to always add extra grid points until there is a grid point at or below the specified level. Positions with high coverage may still have extra grid points added beyond this level. Using this option may result in extra low-AF calls being made, but note that it also causes low-AF noise hypotheses to be considered, which can have the opposite effect (suppression of calls).</p>									

Sample-specific NTD Error Bias Estimation

DRAGEN can compensate for oxidation and deamination artifacts that might exist upstream of the sequencing system, and are common in FFPE samples. DRAGEN does this by estimating nucleotide mutation biases on a per sample basis, taking account of read orientation. During variant calling, DRAGEN corrects for nucleotide substitution biases by combining the estimated parameters with the base call quality scores, modifying the nucleotide error rates used by the hidden Markov model.

Nucleotide (NTD) Error Bias Estimation is on by default and recommended as a replacement for the orientation bias filter. Both methods take account of strand-specific biases (systematic differences between F1R2 and F2R1 reads). In addition, NTD error estimation accounts for non-strand-specific biases such as sample-wide elevation of a certain SNV type, eg C->T or any other transition or transversion. NTD error estimation can also capture the biases in a trinucleotide context.

This feature can be disabled by specifying `--vc-enable-unequal-ntd-errors=false` or set to auto-detect by specifying `--vc-enable-unequal-ntd-errors=auto`. In auto-detect mode, DRAGEN will run the estimation and then disable the use of the estimated parameters if it determines that the sample does not exhibit nucleotide error bias. When the feature is enabled, DRAGEN will estimate a smaller set of parameters in a monomer context by default. To estimate a larger set of parameters in a trimer context (recommended on sufficiently large panels when coverage is above 1000X), specify `--vc-enable-trimer-context=true`.

To specify the regions from which to estimate nucleotide substitution biases, use `--vc-snp-error-cal-bed`. Alternatively, if `--vc-target-bed` is used to specify the target regions for variant calling, and the total bed regions are sufficiently small (maximum 4 megabases), `--vc-snp-error-cal-bed` can be omitted and DRAGEN will use the target bed file for bias estimation. Otherwise, DRAGEN will use a default bed file selected to match the reference, and covering a mixture of coding and noncoding regions.

DRAGEN requires a panel size of at least 150kbp to correctly estimate nucleotide mutation biases when using trimer context, or at least 10kbp when using monomer context. If this requirement is not met for trimer context, DRAGEN falls back on the monomer model, and if it is not met for monomer context, DRAGEN turns the bias estimation feature off.

Unique Molecular Identifiers Support

To optimize variant calling for different UMI use cases, use the following two batch modes. Both options are false by default. Enable UMI-aware variant calling by setting one of them to true.

- `--vc-enable-umi-solid`—The VC UMI solid mode is optimized for solid tumors with post collapsed coverage rates of ~200—300X and target allele frequencies of 5% and higher.
- `--vc-enable-umi-liquid`—The liquid biopsy pipeline is not equivalent to liquid tumor mode. The liquid biopsy pipeline starts from a regular blood sample and looks for low VAF somatic variants from tumor cell free DNA floating in the blood. This type of test enables tumor profiling (diagnosis/biomarker identification) from plasma rather than from tissue, which requires an invasive biopsy. The VC UMI liquid mode is optimized for a liquid biopsy pipeline with post collapsed coverage rates of ~2000X and target allele frequencies of 0.1% and higher.

These batch settings do not include the *vc-systematic-noise filter*. DRAGEN recommends adding the filter. For more information, see [Systematic Noise Filtering on page 137](#).

If a third-party tool is used to produce the collapsed reads, then configure the tool so that the base call quality scores quantify the error produced by the sequencing system only. DRAGEN uses the [Sample-specific NTD Error Bias Estimation on page 131](#) to account for errors upstream of the sequencing system, so such errors should not be included in base call quality scores.

Post Somatic Calling Filtering

Options

The following options are available for post somatic calling filtering:

Option	Description
--vc-sq-call-threshold	Emits calls in the VCF. The default is 3.0 for tumor-normal and 0.1 for tumor-only. If the value for <i>vc-sq-filter-threshold</i> is lower than <i>vc-sq-call-threshold</i> , the filter threshold value is used instead of the call threshold value.
--vc-sq-filter-threshold	Marks emitted VCF calls as filtered. The default is 17.5 for tumor-normal and 3.0 for tumor-only.
--vc-enable-triallelic-filter	Enables the multiallelic filter. The default is true.
--vc-enable-non-primary-allelic-filter	Similar to the triallelic filter, but filters less aggressively. Keep the allele per multiallelic position with highest alt AD, and only filter the rest (Default=false). Cannot be enabled when the triallelic filter is also on.
--vc-enable-af-filter	Enables the allele frequency filter for nuclear chromosomes. The default value is false. When set to true, the VCF excludes variants with allele frequencies below the AF call threshold or variants with an allele frequency below the AF filter threshold and tagged with low AF filter tag. The default AF call threshold is 1% and the default AF filter threshold is 5%. To change the threshold values, use the following command line options: <i>vc-af-call-threshold</i> and <i>vc-af-filter-threshold</i> . Use <i>vc-enable-af-filter-mito</i> and the corresponding threshold options for mitochondrial allele frequency filtering.
--vc-enable-non-homref-normal-filter	Enables the non-homref normal filter. The default value is true. When set to true, the VCF filters out variants if the normal sample genotype is not a homozygous reference.

Option	Description
--vc-enable-vaf-ratio-filter	Adds one condition to be filtered out by the <code>alt_allele_in_normal</code> filter. The default value is false. When set to true, the VCF filters out variants if the normal sample AF is greater than 20% of tumor sample AF.
--vc-depth-filter-threshold	Filters all somatic variants (alt or homref) with a depth below this threshold. The default value is 0 (no filtering).
vc-homref-depth-filter-threshold	In gVCF mode, filters all somatic homref variants with a depth below this threshold. The default value is 3.
vc-depth-annotation-threshold	Filters all non-PASS somatic alt variants with a depth below this threshold. The default value is 0 (no filtering).

Filters

The following table describes the available somatic calling filters.

Somatic Mode	Filter ID	Description
Tumor-Only & Tumor-Normal	weak_evidence	Variant does not meet likelihood threshold. The likelihood ratio for SQ tumor-normal is < 17.5 or < 3.0 for SQ tumor-only.
Tumor-Only & Tumor-Normal	multiallelic	Site filtered if there are two or more ALT alleles at this location in the tumor.
Tumor-Only & Tumor-Normal	str_contraction	Suspected PCR error where the ALT allele is one repeat unit less than the reference. Enabled only when using <code>-vc-enable-gatk-acceleration=true</code> .
Tumor-Only & Tumor-Normal	base_quality	Median base quality of ALT reads at this locus is < 20.
Tumor-Only & Tumor-Normal	mapping_quality	Median mapping quality of ALT reads at this locus is < 20 (tumor-normal) or < 30 (tumor-only).
Tumor-Only & Tumor-Normal	fragment_length	Absolute difference between the median fragment length of alt reads and median fragment length of ref reads at a given locus > 10000.

Somatic Mode	Filter ID	Description
Tumor-Only & Tumor-Normal	read_position	Median of distances between the start and end of read and a given locus < 5 (the variant is too close to edge of all the reads).
Tumor-Only & Tumor-Normal	low_af	Allele frequency is below the threshold specified with --vc-af-filter-threshold (default is 5%). Enabled only when using --vc-enable-af-filter=true.
Tumor-Only & Tumor-Normal	systematic_noise	If AQ score is < 10 (default) for tumor-normal or < 60 (default) for tumor-only, the site is filtered.
Tumor-Only & Tumor-Normal	low_frac_info_reads	The fraction of informative reads (denominator excludes filtered_out reads) is below the threshold. The default threshold value is 0.5.
Tumor-Only & Tumor-Normal	low_depth	The site was filtered because the number of reads is too low. The filter is off by default.
Tumor-Only & Tumor-Normal	low_tlen	The site was filtered because the fraction of low TLEN ALT supporting reads is above a threshold. The default threshold is 0.4. Reads with TLEN smaller than -2.25 (default) standard deviations from the mean are considered to be low TLEN. This filter is not applied for reads sampled from tight insert distributions ie, stddev / mean < 0.1 (default)

Tumor-Normal and Tumor-Only

Somatic Mode	Filter ID	Description
Tumor-Normal	noisy_normal	More than three alleles are observed in the normal sample at allele frequency above 9.9%.
Tumor-Normal	alt_allele_in_normal	ALT allele frequency in the normal sample is above 0.2 plus the maximum contamination tolerance. For solid tumor mode, the value is 0. For liquid tumor mode, the default value is 0.15. See <code>vc-enable-vaf-ratio-filter</code> for optional conditions.

Somatic Mode	Filter ID	Description
Tumor-Normal	filtered_reads	More than 90% of reads have been filtered out.
Tumor-Normal	no_reliable_supporting_read	No reliable supporting read was found in the tumor sample. A reliable supporting read is a read supporting the alt allele with mapping quality ≥ 40 , fragment length $\leq 10,000$, base call quality ≥ 25 , and distance from start/end of read ≥ 5 .
Tumor-Normal	too_few_supporting_reads	Variant is supported by < 3 reads in the tumor sample.
Tumor-Normal	non_homref_normal	Normal sample genotype is not a homozygous reference.
Tumor-Normal	germline_risk	Likelihood of allele being present in normal > 0.025 . Enabled only when using <code>--vc-enable-gatk-acceleration=true</code> .
Tumor-Normal	artifact_in_normal	TLOD of the normal read set (Normal artifact LOD) is > 0.0 . Not called normal artifact if allele fraction in normal is much smaller than allele fraction in tumor ($\text{normalAlleleFraction} < (0.1 * \text{tumorAlleleFraction})$). Enabled only when using <code>--vc-enable-gatk-acceleration=true</code> .

QUAL is not output in the somatic variant records. Instead, the confidence score is FORMAT/SQ.

```
##FORMAT<ID=SQ,Number=1>Type=Float,Description="Somatic quality">
```

The field is specific to the sample. For the tumor samples, the field quantifies the evidence that a somatic variant is present at a given locus.

If a normal sample is also available, the corresponding FORMAT/SQ value quantifies the evidence that the normal sample is a homozygous reference at a given locus.

GQ is not output in the somatic variant records, because DRAGEN does not test for multiple diploid genotype candidates. Instead, an ALT allele is considered as a candidate somatic variant. If tumor SQ $> \text{vc-sq-call-threshold}$ (default is 3), then the FORMAT/GT for the tumor sample is hard-coded to 0/1, and the FORMAT/AF yields an estimate on the somatic variant allele frequency, which ranges anywhere within [0,1].

- If tumor SQ $< \text{vc-sq-call-threshold}$, the variant is not emitted in the VCF.
- If tumor SQ $> \text{vc-sq-call-threshold}$ but tumor SQ $< \text{vc-sq-filter-threshold}$, the variant is emitted in the VCF, but FILTER=weak_evidence.

- If tumor SQ > vc-sq-call-threshold and tumor SQ > vc-sq-filter-threshold, the variant is emitted in the VCF and FILTER=PASS (unless the variant is filtered by a different filter).
- The default vc-sq-filter-threshold is 17.5 for tumor-normal and 3.0 for tumor-only analysis.

The following is an example somatic T/N VCF record. Tumor SQ > vc-sq-call-threshold but tumor SQ < vc-sq-filter-threshold, so the FILTER is marked as weak_evidence.

```
2 593701 . G A . weak_evidence
DP=97;MQ=48.74;SQ=3.86;NLOD=9.83;FractionInformativeReads=1.000
GT:SQ:AF:F1R2:F2R1:DP:SB:MB 0/0:9.83:33,0:0.000:14,0:19,0:33
0/1:3.86:61,3:0.047:29,2:32,1:64:35,26,0,3:39,22,1,2
```

The clustered-events penalty is an exception to the above rule for emitting variants. By default, the clustered-events penalty replaces the clustered-events filter. Instead of applying a hard filter when too many events are clustered together, DRAGEN applies a penalty to the SQ scores of cophased clustered events. Clustered events with weak evidence are no longer called, but clustered events with strong evidence can still be called. This is equivalent to lowering the prior probability of observing clustered cophased variants. The penalty is applied after the decision to emit variants, so that penalized variants still appear in the VCF if their unpenalized score is high enough. Variants that are combined into an MNV via the --combine-phased-variants-distance option are treated as a single variant for the purposes of the penalty. To disable the clustered-events penalty, set --vc-clustered-event-penalty=0.

gVCF Output

You can output a gVCF file for tumor-only data sets. A gVCF file reports information on every position of the input genome, including homozygous reference (homref) positions, ie positions where no alt allele (either germline or somatic) is present. DRAGEN creates a new <NON_REF> allele, to which reads that do not support the reference allele or any reported variant allele are assigned. In tumors, variants could exist at arbitrarily low allele frequencies and be undetectable. Thus, a somatic homref call cannot guarantee that no somatic variant at any allele frequency exists at the position. Instead, DRAGEN considers a position to be a homozygous reference if there are no somatic variants with an allele frequency at or above the limit of detection (LOD). Whereas the SQ score for an ordinary alt allele is a phred-scale posterior probability, the SQ score for the <NON_REF> allele is a phred-scale ratio between the likelihood of a homref call and the likelihood of a variant call with allele frequency at the LOD (if an alt allele is also reported, the <NON_REF> SQ score is capped at the complement of the posterior probability for the alt allele). If the LOD value is lowered, fewer homref calls are made. If the LOD value is increased, more homref calls are made.

By default the LOD is set to 5%, but you can enter a different value using the --vc-gvcf-homref-lod option.

Systematic Noise Filtering

The DRAGEN systematic noise filter is available in somatic mode, and can be used to reduce false positive calls by accounting for site-specific noise. This filter replaces the `panel of normals` option. Unlike the panel of normals filter that blocked certain positions, the systematic noise filter uses a statistical model. The systematic noise at each position is estimated by extracting the variant call allele frequencies at the same position from the normal samples and computing the mean or max allele frequency. During the somatic run variant calls are not filtered if the variant call's allele frequency is statistically much higher than the estimated noise at the same position. The filter is considered essential for tumor-only runs where a matched normal is not available, and is also recommended in tumor-normal mode.

During the construction of the noise file DRAGEN aims to detect germline calls and not include them as noise. The recommended strategy to identify germline variants in the normal samples is enabled by `--vc-enable-germline-tagging=true` along with the required Nirvana settings. To explicitly skip this step when generating the normal VCFs please enable `vc-skip-germline-tagging=true` along with an optional AF cutoff for `build-sys-noise-germline-vaf-threshold`. DRAGEN can estimate the noise as either the mean or max AF of all the normal variants at a location.

To enable the systematic noise filter during somatic variant calling use the option `--vc-systematic-noise {NOISE_FILE_PATH}`. For each site a P-value test will be conducted to assess whether a somatic variant may be explained by the noise model. The systematic noise uses a binomial model where the null hypothesis assumes that the variant's alt supporting reads can be explained by the observed systematic noise. If the variant call has an allele frequency that is significantly higher than the systematic noise, then the P value will be low, indicating that the null hypothesis can be discarded and the variant treated as a real variant rather than noise.

Step 1: Option to Generate a custom Systematic Noise BED file

The systematic noise file is generated from normal samples. When possible, it is recommended to build systematic noise files that are library prep, sequencing system, and panel specific. It is especially true for UMI samples that tend to have less noise than WGS/WES samples. Using a general WES/WGS noise file on a clean UMI sample will result in overly aggressive filtering. For noise generation it is recommended to use approximately 20-50 normal samples. Fewer normal samples (1-10) can still be used to generate useful noise files.

The BaseSpace Sequence Hub DRAGEN CNV Baseline Builder App can be used to build noise files in the cloud.

Step 1.a

First run DRAGEN somatic tumor-only on each of approximately 20-50 normal samples with `--vc-detect-systematic-noise=true`. This setting will also require `--vc-enable-germline-tagging=true`, unless explicitly disabled with `--vc-skip-germline-tagging=true`. Run DRAGEN in somatic mode by specifying inputs with `--tumor-fastq1`, `--tumor-fastq2` or `--tumor-bam-input`. When building UMI noise files the option `--vc-detect-systematic-noise=true` can also be replaced by `--vc-enable-umi-solid true` or `--vc-enable-umi-liquid true`.

i | `--vc-detect-systematic-noise` should only be used when running the small variant caller for systematic noise estimation. This setting is optimized to detect small amounts of noise and is not intended for processing tumor samples.

The following is an example command line used to generate the normal VCFs:

```
dragen \
-r {REFERENCE} \
--tumor-bam-input {NORMAL_BAM} OR --tumor-fastq-list {NORMAL_FASTQ_LIST} \
--tumor-fastq-list-sample-id ${NORMAL_FASTQ_LIST_SAMPLE_ID} \
--enable-variant-caller=true \
--vc-detect-systematic-noise=true \
--build-sys-noise-germline-vaf-threshold=1 \
--vc-enable-germline-tagging=true \
--enable-variant-annotation=true \
--variant-annotation-data {NIRVANA_ANNOTATION_FOLDER} \
--variant-annotation-assembly {REFERENCE} \
--output-directory {DIR} \
--output-file-prefix {PREFIX}
```

Step 1.b

Use the normal VCFs from Step 1.a and aggregate the results in order to construct the systematic noise file:

Option	Description
<code>--build-sys-noise-vcfs-list</code>	Text file containing the paths of normal VCFs. Specify the full VCF file paths. List one file per line.
<code>--build-sys-noise-germline-vaf-threshold</code>	Variant calls with VAF higher than this threshold will be considered germline and will not contribute to the noise estimate. This option is disabled by default by setting the threshold to 1. (Default 1)
<code>--build-sys-noise-use-germline-tag</code>	This option will ensure that variants tagged by <code>vc-enable-germline-tagging=true</code> will not be counted as noise.

Option	Description
--build-sys-noise-method	Method to calculate the systematic noise level across samples. Valid options are "mean" and "max". When using the "mean" method noise is calculated as (total ALT AD)/(total DP) per loci. The "max" method calculates the noise per location as the highest VAF observed for any of the samples spanning the location. The "mean" method favors variant calling sensitivity while the "max" method is more aggressive and may be used to improve specificity. The "max" method will generally work well on WGS runs, while "mean" may be more appropriate for smaller exomes or panels. When building a custom noise file, users are encouraged to try both settings. (Default "mean")
--build-sys-noise-decimal-precision	Number of decimal digits emitted in noise file. Options are [3-6]. This option should only be required when building very large noise files and if it is desired to keep the file size all small as possible. The default option of 5 has sufficient accuracy for all samples (including sensitive UMI samples). For larger samples, e.g. WGS with 50-500X coverage, 3 decimal places could be sufficient and this setting may potentially be used to help reduce the noise file size. (Default 5)
--build-sys-noise-threads	Number of threads used to generate the noise file. Each thread consumes approximately 55GB of memory. The default setting of 2 threads should work well on servers with more than 128GB of system memory. (Default 2)

Option	Description
--build-sys-noise-min-sample-cov	Min coverage at a site for a sample to be used towards noise estimation. At low coverages estimated allele frequencies become less reliable. Accurate AF estimation is important for germline variant detection, and also for noise detection when using MAX noise. (Default 5)
--build-sys-noise-min-supporting-samples	Min number of samples with noise at a position in order for a position to be considered systematic-noise (Default 1).

Step 1: Alternative option - use prebuilt Systematic Noise BED Files

The following prebuilt systematic noise files for WGS and WES are available for download on the [Illumina DRAGEN Bio-IT Platform support site](#) files page.

Version	DRAGEN Release	Modes	Normal Samples
Somatic Systematic Noise Baseline Collection v1.0.0	DRAGEN 3.7	hg19, hg38, hs37d5, Nextera, TruSeq, WES, WGS	20-50 per cohort, 80-100X coverage
Somatic Systematic Noise Baseline Collection v1.0.1	DRAGEN 4.2	hg19, hg38, hs37d5, Nextera, TruSeq, WES, WGS	~50 per cohort, 80-100X coverage

The WGS noise files are generated using a combination of Nextera and TruSeq (with and without PCR), while dedicated Nextera and TruSeq WES noise files are available. Each noise file is generated with the "mean" and "max" noise extraction methods. Use the "mean" noise files for higher sensitivity, or "max" for higher specificity.

The v1.0.1 noise files used Nirvana germline annotation to avoid counting germline calls as noise. The new noise files are expected to better preserve somatic sensitivity in regions where germline calls are common.

Step 2 : Running DRAGEN somatic variant caller with a systematic noise filter.

When running DRAGEN T/O or T/N somatic pipelines it is recommend to use the systematic noise filter. The systematic noise file is used to compute P-values for any somatic variants. Under the null hypothesis a variant can be explained under the binomial noise model, where the mean noise for each site is obtained from the systematic noise file. A P-value is computed, Phred-scaled, and then represented as AQ score. If the systematic noise AQ score is smaller than the defined threshold, the variant is filtered as systematic noise.

The following systematic noise command line options are available:

Command	Description
--vc-systematic-noise	Specifies a systematic noise BED file. If a somatic variant does not pass the AQ threshold, the variant is marked as 'systematic_noise' in the FILTER column of the output VCF.
--vc-systematic-noise-filter-threshold	Set the AQ threshold. Higher values filter more aggressively. By default the threshold value for tumor-normal is 10 and 60 for tumor-only. The valid range spans 0-100. For tumor-normal runs the threshold may be set higher (e.g to 60) to improve specificity at the possible cost of some sensitivity.

Germline Tagging in Tumor-Only Pipeline

When enabling DRAGEN for tumor-only somatic calling, potential germline variants can be tagging in the INFO field with GermlineStatus using population databases. The following options are available for this feature:

Option	Description
--vc-enable-germline-tagging	Enables germline tagging. The default is false. Once this is set to true it will require user to set annotation related parameters as follows: --enable-variant-annotation=true --variant-annotation-data Nirvana annotation database. For more information see Download Data Files . --variant-annotation-assembly Genomic build, eg GRCh37.

Additional options to control how to define germline variants.

Option	Description
--germline-tagging-db-threshold	The minimum alternative allele count in population database for a variant to be defined as germline. The default=50.
--germline-tagging-pop-af-threshold	The minimum population allele frequency for a variant to be defined as germline. Once specified, this will override the input from --germline-tagging-db-threshold.

```
1 11301714 . A G . PASS
DP=3626;MQ=249.61;FractionInformativeReads=0.974;AQ=100.00;GermlineStatus=Germline_DB
GT:SQ:AD:AF:F1R2:F2R1:DP:SB:MB
0/1:64.73:1772,1758:0.498:872,901:900,857:3530:846,926,843,915:894,878,874,884
```

Joint Analysis for Multiple Samples

DRAGEN supports pedigree-based and population-based germline variant joint analysis for multiple samples. In pedigree-based analysis, samples from the same species are related to each other. In population-based analysis, samples of the same species are unrelated to each other.

Joint analysis requires a gVCF file for each sample. To create a gVCF file, run the germline small variant caller with the `--vc-emit-ref-confidence gVCF` option. There is also the option to write a germline gVCF with reduced size using the option `--vc-compact-gvcf`. This results in a significant speed up for a downstream analysis using gVCF Genotyper. However, this format is not compatible with a pedigree analysis.

The gVCF file contains information on the variant positions and positions determined to be homozygous to the reference genome. For homozygous regions, the gVCF file includes statistics that indicate how well reads support the absence of variants or alternative alleles. Contiguous homozygous runs of bases with similar levels of confidence are grouped into blocks, referred to as hom-ref blocks. Not all entries in the gVCF are contiguous. A reference might contain gaps that are not covered by either variant line or a hom-ref block. Gaps correspond to regions that are not callable. A region is not callable if there is not at least one read mapped to the region with a MAPQ score above zero. The following example shows a joint VCF where one sample has a variant, and the other two samples are in a gVCF gap. Gaps are represented by ".:/:::".

```
1 605262 . G A 13.41 DRAGENHardQUAL
AC=2;AF=1.000;AN=2;DP=2;FS=0.000;MQ=14.00;QD=6.70;SOR=0.693
GT:AD:AF:DP:GQ:FT:F1R2:F2R1:PL:GP .:/::::::::::LowDepth
1/1:0,2:1.000:2:4:PASS:0,0:0,2:50,6,0:1.383e+01,4.943e+00,1.951e+00
./::::::::::LowDepth
```

Combined phased variants in the gVCF input

The DRAGEN germline variant caller has an option `--vc-combine-phased-variants-distance` to combine phased variants in the gVCF output. Input gVCF files created with this option cannot be processed in a population-based analysis using gVCF Genotyper.

The option to combine phased variants is switched off by default, for details please refer to the section on germline small variant calling in this user guide.

Force-genotyped and targeted variants in the gVCF input

If force genotyping was enabled for any input file, any ForceGT calls that are not also called by the variant caller will be ignored.

Similarly, targeted variant calls (option `--targeted-merge-vc`) in any gVCF file that are not also called by the variant caller will be ignored as well.

Processing GATK gVCF Files

Both pedigree- and population-based joint analysis can process gVCF files written by the GATK v4.1 variant caller, by using the command line option `--vc-enable-gatk-acceleration=true`.

Joint Analysis Output Format

There are two available joint analysis output files:

File	Description
Multisample VCF	A VCF file containing a column with genotype information for each of the input samples according to the input variants.
Multisample gVCF	A gVCF file augmenting the content of a multisample VCF file, similar to how a gVCF file augments a VCF file for a single sample. In between variant sites, the multisample gVCF contains statistics that describe the level of confidence that each sample is homozygous to the reference genome. Multisample gVCF is a convenient format for combining the results from a pedigree or small cohort into a single file. If using a large number of samples, fluctuation in coverage or variation in any of the input samples creates a new hom-ref block, which causes a highly fragmented block structure and a large output file that can be slow to create. This is only available in the pedigree-based analysis.

Hom-ref Blocks FORMAT Fields

In hom-ref blocks, the following FORMAT fields are calculated uniquely.

Field	Description
FORMAT/DP	In a single sample gVCF, the FORMAT/DP reported at a HomRef position is the median DP in that band. In a joint VCF, the FORMAT/DP reported at a HomRef position is the MIN_DP from homref calls.
FORMAT/AD	In single sample gVCF, values represent the position in the band where DP=median DP. In the joint VCF, AD values at HomRef positions are copied from the single sample gVCF.
FORMAT/AF	Values are based on FORMAT/AD.
FORMAT/PL	Values represent the Phred likelihoods per genotype hypothesis. For hom-ref blocks, each value in FORMAT/PL represents the minimum value across all positions within the band.
FORMAT/SPL and FORMAT/ICNT	Parameters reported in the gVCF records, including both hom-ref blocks and variant records. The parameters are used to compute the confidence score of a variant being de novo in the proband of a trio. For SNP, FORMAT/PL and FORMAT/SPL are both used as input to the DeNovo Caller. FORMAT/PL represents Phred likelihoods obtained from the genotyper, if the genotyper is called. FORMAT/SPL represents Phred likelihoods obtained from column-wise estimation, pregraph. Each value in FORMAT/SPL represents the minimum across all positions within the band. For INDEL, the PL value is computed in the joint pedigree calling step based on the FORMAT/ICNT reported in the gVCF file. FORMAT/ICNT consist of two values. The first value is the number of reads with no indels at the position, and the second value is the number of reads with indels at the position. Each value in FORMAT/ICNT represents the maximum of the value across all positions within the band.

In the following example hom-ref block, ICNT provides information on whether each sample contains an Indel at the position of interest. If the proband contains an indel at the position and the ICNT of the parents does not indicate any read supporting an indel, then the confidence score is high for the proband to have an indel de novo call at the position.

```
chr1 10288 . C <NON_REF> . PASS END=10290
GT:AD:DP:GQ:MIN_DP:PL:SPL:ICNT
0/0:131,4:135:69:132:0,69,1035:0,125,255:23,1
chr1 10291 . C
T,<NON_REF> 38.45 PASS
DP=100;MQ=24.72;MQRankSum=0.733;ReadPosRankSum=4.112;FractionInformativeReads=0
.600;R2_5P_bias=0.000
GT:AD:AF:DP:F1R2:F2R1:GQ:PL:SPL:ICNT:GP:PRI:SB:MB
0/1:28,32,0:0.533,0.000:60:20,21,0:8,11,0:15:73,0,12,307,157,464:255,0,255:23,1
0:3.8452e+01,1.3151e-
01,1.5275e+01,3.0757e+02,1.9173e+02,4.5000e+02:0.00,34.77,37.77,34.77,69.54,37.
77:4,24,7,25:8,20,14,18
```

SPL and ICNT values are specific to DRAGEN. The GATK variant caller does not output SPL and ICNT values.

In a single sample gVCF, FORMAT/DP reported at a HomRef position is the median DP in the band. The minimum is also computed and printed as MIN_DP for the band.

In the joint VCF, MIN_DP from homref calls is printed as FORMAT/DP, and AD is just copied from the gVCF. So in the jointVCF Homref position, the DP is not necessarily going be the sum of AD.

Pedigree Mode

Use pedigree mode to jointly analyze samples from related individuals and to perform *de novo* calling.

To invoke pedigree mode, set the --enable-joint-genotyping option to true. Use the --pedigree-file option to specify the path to a pedigree file that describes the relationship between panels.

The pedigree file must be a tab-delimited text file with the file name ending in *.ped. The following information is required.

Column Header	Description
Family_ID	The pedigree identifier.
Individual_ID	The ID of the individual.
Paternal_ID	The ID of the individual's father. If the founder, the value is 0.
Maternal_ID	The ID of the individual's mother. If the founder, the value is 0.

Column Header	Description
Sex	The sex of the sample. If male, the value is 1. If female, the value is 2.
Phenotype	The genetic data of the sample. If unknown, the value is 0. If unaffected, the value is 1. If affected, the value is 2.

The following is an example of an input pedigree file.

#Family_ID	Individual_ID	Paternal_ID	Maternal_ID	Sex	Phenotype
FAM001	NA12877_Father	0	0	1	1
FAM001	NA12878_Mother	0	0	2	1
FAM001	NA12882_Proband	NA12877_Father	NA12878_Mother	2	2
FAM001	NA12883_Proband	NA12877_Father	NA12878_Mother	1	0

De Novo Calling

The De Novo Caller identifies all the trios within the pedigree and generates a *de novo* score for each child. The De Novo Caller supports multiple trios within a single pedigree. Pedigree Mode supports *de novo* calling for small, structural, and copy number variants.

Pedigree Mode is run in multiple steps. The following is an example workflow for a trio using FASTQ input.

1. Run single sample alignment and variant calling to generate per sample output using the following inputs for Pedigree Mode.
 - gVCF files for Small Variant Caller.
 - *.tn.tsv files for the Copy Number Caller.
 - BAM files for the Structural Variant Caller.
 2. Run Pedigree Mode for Small Variant Caller.
- For more information, see [DeNovo_SmallVariant_fDG](#) on page 1.

3. Run Pedigree Mode for Copy Number Caller.
For more information, see [Multisample CNV Calling on page 1](#).
4. Run Pedigree Mode for Structural Variant Caller.
For more information, see [Structural Variant De Novo Quality Scoring on page 317](#).
5. Run DeNovo Variant Small Variant Filtering.
For more information, see [De Novo Small Variant Filtering on page 158](#).

Small Variant De Novo Calling

The Small Variant De Novo Caller considers a trio of samples at a time. The samples are related via a pedigree file. The Small Variant De Novo Caller determines all positions that have a Mendelian conflict based on the genotype from the individual sample gVCFs. Sex chromosomes in males are treated as haploid apart from the PAR regions, which are treated as diploid.

Each of the positions is then processed through the Pedigree Caller to compute a joint posterior probability matrix for the possible genotypes. The probabilities are used to determine whether the proband has a *de novo* variant with a DQ confidence score. All three subjects are assumed to have an independent error probability.

At positions where the original genotype from the gVCFs shows a double Mendelian conflict (eg, 0/0+0/0->1/1 or 1/1+1/1->0/0), the genotypes of the trio samples can be adjusted to the highest joint posterior probability that has at least one Mendelian conflict.

The DQ formula is $DQ = -10\log_{10}(1 - P_{denovo})$.

P_{denovo} is the sum of all indexes in the joint posterior probability matrix with one or more Mendelian conflicts.

In the GT overwrite step, it is possible for the GT of the parents to be overwritten. In the case of multiple trios, the GT of the parents is based on the last trio processed. The trios are processed in the order they are listed in the pedigree file. DRAGEN currently does not add an annotation in the VCF in cases where the GT was overwritten.

The multisample VCF file is annotated with FORMAT/DQ and FORMAT/DN fields to the output a VCF file that represents a *de novo* quality score and an associated *de novo* call. The DN field in the VCF is used to indicate the *de novo* status for each segment.

The following are the possible values:

- Inherited—The called trio genotype is consistent with Mendelian inheritance.
- LowDQ—The called trio genotype is inconsistent with Mendelian inheritance and DQ is less than the *de novo* quality threshold.
- DeNovo—The called trio genotype is inconsistent with Mendelian inheritance and DQ is greater than or equal to the *de novo* quality threshold.

The following example shows the VCF line for a trio:

```
1 16355525. G A 34.46 PASS
```

```
AC=1;AF=0.167;AN=6;DP=45;FS=6.69;MQ=108.04;MQRankSum=0.156;QD=2.46;ReadPosRankSu
m=0;SOR=0.01
GT:AD:AF:DP:GQ:FT:F1R2:F2R1:PL:GP:PP:DPL:DN:DQ
0/1:11,3:0.214:14:39:PASS:8,2:3,1:74,0,47:39.454,0.00053613,49.99:0,1,104:74,0,4
7:DeNovo:0.6737
0/0:18,0:0:16:48:PASS:::0,48,605:::0,12,224:0,48,255:::0/0:14,0:0:14:42:PASS::
::0,42,490:::0,5,223:0,42,255:::
```

Pedigree Mode Options

The following command line options are available for *de novo* small variant calling.

Option	Description
--enable-joint-genotyping	Run the joint genotyping caller.
--pedigree-file	Specify the path to a pedigree file that describes the relationship between samples. It is possible to run JointGenotyper without a pedigree file on unrelated samples, but we do not recommend this anymore for gVCF variant calls from DRAGEN 3.10 or newer.
--variant or --variant-list	Specify the gVCF input to the workflow. The pedigree caller can read input gVCF files from an AWS S3 bucket, Azure storage BLOB, or pre-signed URL.
--qc-snp-denovo-quality-threshold	Specify the minimum DQ value for an SNP to be considered <i>de novo</i> . The default value is 0.05.
--qc-indel-denovo-quality-threshold	Specify the minimum DQ value for an indel to be considered <i>de novo</i> . The default value is 0.02.
--output-directory	The output directory. This is required.
--output-file-prefix	The prefix used to label all output files. This is required.
-r	The directory where the hash table resides.

Population Mode

DRAGEN provides a population-based analysis option to jointly analyze samples from unrelated individuals.

The tool for population-based analysis is the iterative gVCF Genotyper. Its input is a set of single or multisample gVCFs. The output is a multisample VCF that contains one entry for any variant seen in any of the input gVCFs. The variants are genotyped across all input samples using information from the hom-ref blocks as necessary. The iterative gVCF Genotyper does not adjust genotypes based on population information but it provides means to filter variant sites based on information leveraged from the population. See [Iterative gVCF Genotyper analysis](#) for information on the available command line options.

To compare multiple pedigrees, you can run gVCF Genotyper on the output of a pedigree analysis and merge multiple joint-called pedigrees into a single multisample VCF. To enable, run the pedigree analysis using the `--enable-multi-sample-gvcf=true` option to write a multisample gVCF.

Iterative gVCF Genotyper Analysis

Iterative gVCF Genotyper Analysis offers an iterative workflow to aggregate new samples into an existing cohort. The iterative workflow allows users to incrementally aggregate new batches of samples with existing batches, without having to redo the analysis from scratch across all samples every time when new samples are available. The workflow takes single sample gVCF files as input, and can be performed in a step-by-step mode if multiple batches of samples are available, or end-to-end mode, if only a single batch of samples is available. Multi-sample gVCF files output from the Pedigree Caller are also accepted as input.

Workflow steps

1. gVCF aggregation

Users can use iterative gVCF Genotyper to aggregate a batch of gVCF files into a cohort file and a census file. The cohort file is a condensed data format to store gVCF data in multiple samples, similar to a multisample gVCF. The census file stores summary statistics of all the variants and hom-ref blocks among samples in the cohort. When a large number of samples are available, users can divide samples into multiple batches each with a similar sample size (eg 1000 samples), and repeat Step 1 for every batch. If force genotyping was enabled for any input file, any ForceGT calls that are not also called by the variant caller will be ignored.

2. Census aggregation

After all per batch census files are generated, users can aggregate them into a single global census file. This step scales to aggregate thousands of batches, in a much more efficient way compared to aggregating gVCFs from all batches. When a new batch of samples becomes available users only need to perform Step 1 on the batch, aggregate the census file from the batch with the global census file from all previous batches, and generate an updated global census file.

3. msVCF generation

When a global census file is updated with new variant sites discovered and/or variant statistics updated at existing variant sites, the user can take per-batch cohort file, per-batch census file, and the global census file as input, and generate a multisample VCF for one batch of samples. The output multisample VCF contains the variants and alleles discovered in all samples from all batches, and also includes global statistics such as allele frequencies, the number of samples with or without genotypes, and the number of samples without coverage. Similar statistics among samples in the batch are also included. This step can be repeated for every batch of samples, and the number of records in each output multisample VCF is the same across all batches.

To facilitate parallel processing on distributed compute nodes the user can choose to split the genome into shards of equal size for each step, and process each shard using one instance of iterative gVCF Genotyper on each compute node. See option `--shard`.

There is a special treatment of alternative or unaligned contigs when the `--shard` option is enabled: all contigs that are not autosomes, X, Y or chrM are included in the last shard. No other contigs will be assigned to the last shard. The mitochondrial contig will always be on its own in the second to last shard.

If a combined msVCF of all batches is required, an additional step should run separately to merge all of the batch msVCF files into a single msVCF containing all samples.

Command-line arguments common to all steps

<code>--enable-gvcf-genotyper-iterative</code>	Set to true to run the iterative gVCF genotyper (always required).
<code>--ht-reference</code>	The file containing the reference sequence in FASTA format (always required).
<code>--output-directory</code>	The output directory (always required).
<code>--output-file-prefix</code>	The prefix used to label all output files (optional, default value <code>dragen</code>).
<code>--shard</code>	Use this option to process only a portion (shard) of the genome, when distributing the work across multiple compute nodes. Provide the index (1-based) on the shard to process and the total number of shards, in the format n/N (eg 1/50 = shard 1 of 50 shards). To facilitate concurrent processing within each job, the shard will by default be split into 10x the number of available threads.
<code>--gg-regions</code>	<p>if the <code>--shard</code> option is not given, use this option to run iterative gVCF genotyper only for a subset of regions in the genome. The same regions must be used for each step.</p> <ul style="list-style-type: none"> • The value is a list of regions delimited by comma • Each region is processed by one thread on the same compute node

--gg-regions-bed	If a path to a BED file is provided, it will limit the iterative gVCF Genotyper processing to the genome specified regions. This option differs from --gg-regions-bed in that, if the number of regions exceeds 10 times the number of available threads, for example exome data, they will not necessarily be processed by independent threads, making the option faster and compatible with sharding. In this situation there will only take effect on step 1 or end-to-end mode.
--gg-discard-ac-zero	If set to true, the gVCF Genotyper does not print variant alleles that are not called (hom-ref genotype) in any sample. The default value is true.
--gg-remove-nonref	Removes the <NON_REF> symbolic allele from the output of gVCF Genotyper. This option should be used to support downstream tools that cannot process VCF lines with <NON_REF> or to generate more concise msVCFs. This option needs to be enabled in step 1. The default is false.
--gg-vc-filter	Discard input variants that failed filters in the upstream caller. The default is false. Affected records will have their genotype set to hom-ref and the filter string "ggf" added to FORMAT/FT.
--gg-hard-filter	Specifies a filtering expression to be applied to the output msVCF records. See "msVCF hard filtering" below. The default is to apply no filters.
--gg-skip-filtered-sites	Omits msVCF records that fail the given hard filter. The default is false.

Command-line arguments for step 1 (step by step mode)

--gvcfs-to-cohort-census	Set to true to aggregate gVCF files from one batch of samples into a cohort file and a census file.
--variant-list	The path to a file containing a list of input gVCF files, with the path to each file on a separate line.
--variant	if --variant-list is not given, use this option for each input gVCF file.

Command-line arguments for step 2 (step by step mode)

--aggregate-censuses	Set to true to aggregate a list of per batch census files into a global census file.
--input-census-list	The path to a file containing a list of input per batch census files (from Step1), with the path to each file on a separate line.

Command-line arguments for Step 3, step-by-step mode

--generate-msvcf	Set to true to generate a multisample VCF for one batch of samples.
--input-cohort-file	The path to the per batch cohort file (from Step 1).
--input-census-file	The path to the per batch census file (from Step 1).
--input-global-census-file	The path to the global census file (from Step 2).

Command-line arguments for running Steps 1, 2, and 3. End-to-end mode for a single batch

--variant-list	The path to a file containing a list of input gVCF files, with the path to each file on a separate line.
--variant	If --variant-list is not given, use this option for each input gVCF file.

Commandline arguments for merging per-batch msVCF files

--merge-batches	Set to true to merge msVCF files for a set of batches.
--input-batch-list	The path to a file containing a list of msVCF files to be merged, with the path to each file on a separate line. All the files listed must have been generated from the same global census file and all batches pertaining to that global census must be included in the merge.
--gg-enable-indexing	Set to true to generate a tabix index for the merged msVCF (default false).

Enabling Mimalloc for Enhanced Performance

Mimalloc is a custom memory allocation library that can yield significant speed-ups in the iterative gVCF Genotyper workflow. In some deployments, eg cloud, it is automatically and seamlessly used but in other contexts it requires special user intervention to be activated, as at present it cannot be included in standard DRAGEN by default.

The convenience script `mi_dragen.sh` is provided, that loads the bundled library and can be transparently used in the same way as the DRAGEN executable. It is only intended and supported for use with the iterative gVCF Genotyper component, although it can in principle be applied for any other DRAGEN workflow.

 | The use of Mimalloc for other purposes is known to potentially lead to undesirable memory overuse and is not recommended. Using Mimalloc for other purposes is at your own risk.

The multisample VCF output of the iterative gVCF Genotyper

The output of gVCF Genotyper is a multi-sample VCF (msVCF) that contains metrics computed for all samples in the cohort.

The msVCF can become a very large file with increasing cohort size. In some cases, the file might need more storage than can be allocated by VCF parsers. This is caused by VCF entries such as FORMAT/PL which store a value for each combination of alleles. FORMAT/PL was replaced with a tag FORMAT/LPL which stores a value only for the alleles that occur in the sample. The FORMAT/LAA field lists 1-based indices of the alternate alleles that occur in the current sample.

Iterative gVCF Genotyper With Mitochondrial Variant Calls

When processing mitochondrial variant calls, which may contain separate records for each allele, iterative gVCF Genotyper processing differs in the following ways:

- Only the record with the highest FORMAT/AF sum is kept.
- The FORMAT/AF field will be additionally collected, and used to generate the FORMAT/LAF field in the output msVCF

QUAL column in msVCF

The value displayed in the QUAL column of the msVCF is the maximum of the input QUAL values for the site across the global cohort. The QUAL value will be missing if any of the batch census files used to create the global census were generated with a version of DRAGEN earlier than v4.2.

Measures of Hardy-Weinberg Equilibrium in the msVCF output

The Hardy-Weinberg Equilibrium (HWE) states that, given certain conditions, genotype and allele frequencies should remain constant between generations. Deviations from HWE can result from violations of the underlying HWE assumptions in the population, non-random sampling or may be

artifacts of variant calling. Adherence to HWE can be assessed by comparing the observed frequencies of genotypes to those expected under HWE given the observed allele counts.

Iterative gVCF Genotyper (iGG) offers several metrics for assessing adherence to HWE. It calculates both allele-wise and site-wise HWE P-values, an allele-wise excess heterozygosity (ExcHet) P-value and the site-wise inbreeding coefficient (IC). These metrics are calculated only for diploid sites and missing values are excluded from the calculations. These values are included as fields in the INFO column of the output msVCF file. Both batch-wise and global values are included, where the field names for the global values are prefixed with G.

Metric	Description	Scope	Number of values
HWE	Hardy-Weinberg Equilibrium P-value	Allele-wise	One for each alt allele
ExcHet	Excess Heterozygosity P-value	Allele-wise	One for each alt allele
HWEc2	Hardy-Weinberg Equilibrium P-value	Site-wise	1
IC	Inbreeding Coefficient	Site-wise	1

Care should be taken when interpreting these metrics for small cohorts and/or low frequency alleles, as small changes in inputs can lead to large changes in their values. Further, violations of the underlying HWE assumptions (such as inbreeding), and non-random sampling (such as the presence of consanguineous samples), can adversely affect results, making identification of poorly called variants more difficult.

Where it is not possible to calculate the metric, they are represented as missing (ie, ".") in the msVCF file. This can vary between the metrics, but may occur if non-diploid genotypes are encountered, if there is only one allele present at a site, or if no samples are genotyped at a site.

Allele-Wise Hardy-Weinberg Equilibrium and Excess Heterozygosity.

iGG calculates allele-wise HWE and the ExcHet P-values. The values are calculated using the exact-conditional method described in Am J Hum Genet. 2005 May; 76(5): 887–893. The implementation does not use a mid P-value correction.

For HWE a P-value of ≈ 1 suggests that the distribution of heterozygotes and homozygotes is close to that expected under HWE, while a P-value of ≈ 0 suggests a deviation from it. For ExcHet a P-value of ≈ 0.5 suggests that the number of heterozygotes is close to the number expected under HWE, while a value ≈ 1 suggests that there are more heterozygotes than expected and a value ≈ 0 suggests that there are fewer heterozygotes than expected.

For a bi-allelic site the HWE P-values is based on the numbers of homozygotes and heterozygotes comparing the observed to expected. For a multi-allelic site, P-values are calculated per ALT allele as if it were bi-allelic. Genotypes composed of only the ALT allele being considered are counted as

alternative homozygous, any other genotype containing a copy of the ALT allele being considered are counted as a heterozygous, and any genotype with no copies of the ALT allele being considered are counted as reference homozygous (this may include genotypes containing other ALT alleles).

Site-Wise Hardy-Weinberg Equilibrium.

Iterative gVCF Genotyper calculates a site-wise HWE P-value. The value is calculated using the Pearson's chi-squared method, comparing the genotype counts expected under HWE to those observed. The chi-squared test statistic is calculated as:

$$\chi^2 = \sum_{gt} \frac{(E_{gt} - O_{gt})^2}{E_{gt}}$$

where the summation gt is over all genotypes possible at the site given the alleles present, and E_{gt} and O_{gt} are the expected and observed counts for genotype gt respectively. From the chi-squared test statistic the P-value is then calculated from a chi-squared distribution where the number of degrees of freedom is the number of possible genotypes minus the number of alleles, which is

```
dof = n(n-1)/2
```

where n is the number of alleles.

The batch-wise value uses only the alleles present in the batch. Alleles with AC=0 are not included in the calculation.

A P-value of ≈ 1 suggests that the distribution of heterozygotes and homozygotes is close to that expected under HWE, while a P-value of ≈ 0 suggests a deviation from it.

The Inbreeding Coefficient

iGG calculates the inbreeding coefficient (IC) (sometimes called the Fixation index and denoted by F). It is defined as the proportion of the population that is inbred. The value of IC can be estimated by looking at the observed number of heterozygotes in comparison to the number expected under HWE:

```
IC = 1 - O(het) / E(het),
```

where $O(\text{het})$ and $E(\text{het})$ are the observed and expected number of heterozygotes in the cohort, respectively. Although initially conceived for studying inbreeding and defined as a non-negative value, it is also commonly used to look for deviations from HWE and can take values in the range [-1, 1].

Values of IC ≈ 0 suggest that the cohort is in HWE. Negative values suggest an excess of heterozygosity and a deviation from HWE, which can be symptomatic of poor variant calling. Positive values suggest a deficit of heterozygotes and the possible presence of inbreeding.

Using the above definition, IC should be a property of the population, and so would be expected to be drawn from the same distribution for all sites and for all variants at a site. Deviations from this distribution can suggest issues in calling a site correctly. Violations of HWE assumptions and/or non-random sampling may adversely affect the distribution of IC, causing it to be shifted. However, outliers can still be identified, although thresholds may need to be adjusted accordingly.

msVCF hard filtering

Sites in the output msVCF can be filtered on the following global metrics:

- QUAL
- Number of samples with called genotypes (GNS_GT)
- Inbreeding coefficient (GIC)
- X² Hardy-Weinberg Equilibrium P-value (GHWEc2)

The syntax of a filtering expression is the same as that used by the small variant caller. See [Germline Variant Small Hard Filtering on page 159](#). Filters are always applied to the globally-computed metrics, not the values for the current batch. Records failing filter will have the specified filter ID(s) written to the FILTER column of the msVCF, or will be omitted entirely if the `--gg-skip-filtered-sites` option is specified. Because filtering is on a per-site basis, filters cannot be applied separately to SNPs or indels as they can in the variant caller.

De Novo Small Variant Filtering

The filtering step identifies *de novo* variants calls of the joint calling workflow in regions with ploidy changes. Since *de novo* calling can have reduced specificity in regions where at least one of the pedigree members shows non-diploid genotypes, the *de novo* variant filtering marks relevant variants and thus can improve specificity of the call set.

Based on the structural and copy number variant calls of the pedigree, the FORMAT/DN field in the proband is changed from the original DeNovo value to DeNovoSV or DeNovoCNV if the *de novo* variant overlaps with a ploidy-changing SV or CNV, respectively. All other variant details remain unchanged, and all variants of the input VCF will also be present in the filtered output VCF. Structural or copy number variants which result in no change of ploidy, such as inversions, are not considered in the filtering. As an example, a *de novo* SNV calls in the input VCF

```
chr1 234710899 . T C 44.74 PASS
AC=1;AF=0.167;AN=6;DP=73;FS=4.720;MQ=250.00;MQRankSum=5.310;QD=1.15;ReadPosRankS
um=1.366;SOR=0.251
GT:AD:AF:DP:GQ:FT:F1R2:F2R1:PL:GL:GP:PP:DQ:DN
0/1:21,18:0.462:39:48:PASS:14,10:7,8:84,0,50:-8.427,0,-5:4.950e+01,7.041e-
05,5.300e+01:15,0,120:3.2280e-01:DeNovo
0/0:13,0:0.000:11:30:PASS:::::0,30,450::::10,0,227
0/0:25,0:0.000:22:60:PASS:::::0,60,899:::::0,33,227
```

Overlapping with an SV duplication in the proband, mother or father would be represented in the filtered output VCF as follows:

```
chr1 234710899 . T C 44.74 PASS
AC=1;AF=0.167;AN=6;DP=73;FS=4.720;MQ=250.00;MQRankSum=5.310;QD=1.15;ReadPosRankS
um=1.366;SOR=0.251
GT:AD:AF:DP:GQ:FT:F1R2:F2R1:PL:GL:GP:PP:DQ:DN
0/1:21,18:0.462:39:48:PASS:14,10:7,8:84,0,50:-8.427,0,-5:4.950e+01,7.041e-
05,5.300e+01:15,0,120:3.2280e-01:DeNovoSV
0/0:13,0:0.000:11:30:PASS:::::0,30,450::::10,0,227
0/0:25,0:0.000:22:60:PASS:::::0,60,899:::::0,33,227
```

The following is an example command line for running the *de novo* filtering, based on the files returned by the joint calling workflows:

```
dragen \
--dn-enable-denovo-filtering true \
--dn-input-joint-vcf <JOINT_SMALL_VARIANT_VCF> \
--dn-output-joint-vcf <OUTPUT_VCF> \
--dn-sv-vcf <JOINT_SV_VCF> \
--dn-cnv-vcf <JOINT_CNV_VCF> \
--enable-map-align false
```

De Novo Small Variant Filtering Options

The following options are used for de novo variant filtering:

Option	Description
--dn-input-vcf	Joint small variant VCF from the de novo calling step to be filtered.
--dn-output-vcf	File location to which the filtered VCF should be written. If not specified, the input VCF is overwritten.
--dn-sv-vcf	Joint structural variant VCF from the SV calling step. If omitted, checks with overlapping structural variants are skipped.
--dn-cnv-vcf	Joint structural variant VCF from the CNV calling step. If omitted, checks with overlapping copy number variants are skipped.

Germline Variant Small Hard Filtering

DRAGEN provides post-VCF variant filtering based on annotations present in the VCF records. Default and non-default variant hard filtering are described below. However, due to the nature of DRAGEN's algorithms, which incorporate the hypothesis of correlated errors from within the core of variant caller, the pipeline has improved capabilities in distinguishing the true variants from noise, and therefore the dependency on post-VCF filtering is substantially reduced. For this reason, the default post-VCF filtering in DRAGEN is very simple.

Default Small Variant Hard Filtering

The default filters in the germline pipeline are as follows:

- ##FILTER=<ID=DRAGENSnpHardQUAL,Description="Set if true:QUAL < 10.41 (3 when ML recalibration is enabled)">
- ##FILTER=<ID=DRAGENIndelHardQUAL,Description="Set if true:QUAL < 7.83 (3 when ML recalibration is enabled)">
- ##FILTER=<ID=LowDepth,Description="Set if true:DP <= 1">
- ##FILTER=<ID=PloidyConflict,Description="Genotype call from variant caller not consistent with chromosome ploidy">
- DRAGENSnpHardQUAL and DRAGENIndelHardQUAL: For all contigs other than the mitochondrial contig, the default hard filtering consists of thresholding the QUAL value only. A different default QUAL threshold value is applied to SNP and INDEL.
- LowDepth: This filter is applied to all variants calls with INFO/DP <= 1/.
- PloidyConflict: This filter is applied to all variant calls on chrY of a female subject, if female is specified on the DRAGEN command line, or if female is detected by the Ploidy Estimator.

DRAGEN processes it through a continuous AF pipeline for the mitochondrial contig. It is similar to the somatic variant calling pipeline. Refer to [Mitochondrial Calling on page 122](#) for the filtering details.

Non-Default Small Variant Hard Filtering

DRAGEN supports basic filtering of variant calls as described in the VCF standard. You can apply any number of filters with the `--vc-hard-filter` option, which takes a semicolon-delimited list of expressions, as follows:

`<filter ID>:<snp|indel|all>:<list of criteria>`,

where the list of criteria is itself a list of expressions, delimited by the `||` (OR) operator in this format:

`<annotation ID> <comparison operator> <value>`

The meaning of these expression elements is as follows:

- filterID—The name of the filter, which is entered in the FILTER column of the VCF file for calls that are filtered by that expression.
- .snp/indel/all—The subset of variant calls to which the expression should be applied.
- annotation ID—The variant call record annotation for which values should be checked for the filter. Supported annotations include FS, MQ, MQRankSum, QD, and ReadPosRankSum.
- comparison operator—The numeric comparison operator to use for comparing to the specified filter value. Supported operators include `<`, `\leq` , `$=$` , `\neq` , `\geq` , and `$>$` .

For example, the following expression would mark with the label "SNP filter" any SNPs with `FS < 2.1` or with `MQ < 100`, and would mark with "indel filter" any records with `FS < 2.2` or with `MQ < 110`:

```
--vc-hard-filter="SNP filter:snp:FS < 2.1 || MQ < 100; indel filter:indel:FS < 2.2 || MQ < 110"
```

This example is for illustration purposes only and is NOT recommended for use with DRAGEN V3 output. Illumina recommends using the default hard filters.

The only supported operation for combining value comparisons is OR, and there is no support for arithmetic combinations of multiple annotations. More complex expressions may be supported in the future.

Modeling of Correlated Errors Across Reads

DRAGEN has two algorithms that model correlated errors across reads in a given pileup.

Foreign Read Detection

Foreign read detection (FRD) detects mismapped reads. FRD modifies the probability calculation to account for the possibility that a subset of the reads were mismapped. Instead of assuming that mapping errors occur independently per read, FRD estimates the probability that a burst of reads is mismapped, by incorporating such evidence as MAPQ and skewed AF.

Mapping errors typically occur in bursts, but treating mapping errors as independent error events per read can result in high confidence scores in spite of low MAPQ and/or skewed AF. One possible strategy to mitigate overestimation of confidence scores is to include a threshold on the minimum MAPQ used in the calculation. However, this strategy can discard evidence and result in false positives.

FRD extends the legacy genotyping algorithm by incorporating an additional hypothesis that reads in the pileup might be foreign reads (ie, their true location is elsewhere in the reference genome). The algorithm exploits multiple properties (skewed allele frequency and low MAPQ) and incorporates this evidence into the probability calculation.

Sensitivity is improved by rescuing FN, correcting genotypes, and enabling lowering of the MAPQ threshold for incoming reads into the variant caller. Specificity is improved by removing FP and correcting genotypes.

Base Quality Dropoff

The base quality drop off (BQD) algorithm detects systematic and correlated base call errors caused by the sequencing system. BQD exploits certain properties of those errors (strand bias, position of the error in the read, base quality) to estimate the probability that the alleles are the result of a systematic error event rather than a true variant.

Bursts of errors that occur at a specific locus have distinct characteristics differentiating them from true variants. The base quality drop off (BQD) algorithm is a detection mechanism that exploits certain properties of those errors (strand bias, position of the error in the read, low mean base quality over said subset of reads at the locus of interest) and incorporates them into the probability calculation.

Orientation Bias Filter

The orientation bias filter is designed to reduce noise typically associated with the following:

- Pre-adapter artifacts introduced during genomic library preparation (eg, a combination of heat, shearing, and metal contaminates can result in the 8-oxoguanine base pairing with either cytosine or adenine, ultimately leading to G→T transversion mutations during PCR amplification), or
- FFPE (formalin-fixed paraffin-embedded) artifact. FFPE artifacts stem from formaldehyde deamination of cytosines, which results in C to T transition mutations.

The orientation bias filter can only be used on somatic pipelines. To enable the filter, set the `--vc-enable-orientation-bias-filter` option to true. The default is false.

The artifact type to be filtered can be specified with the `--vc-orientation-bias-filter-artifacts` option.

The default is C/T,G/T, which correspond to OxoG and FFPE artifacts. Valid values include C/T, or G/T, or C/T,G/T,C/A.

An artifact (or an artifact and its reverse compliment) cannot be listed twice. For example, C/T,G/A is not valid, because C→G and T→A are reverse compliments.

The orientation bias filter adds the following information.

- `##FORMAT=<ID=F1R2,Number=R,Type=Integer,Description="Count of reads in F1R2 pair orientation supporting each allele">`

- `##FORMAT=<ID=F2R1,Number=R,Type=Integer,Description="Count of reads in F2R1 pair orientation supporting each allele">`
- `##FORMAT=<ID=OBC,Number=1,Type=String,Description="Orientation Bias Filter base context">`
- `##FORMAT=<ID=OBPa,Number=1,Type=String,Description="Orientation Bias prior for artifact">`
- `##FORMAT=<ID=OBParc,Number=1,Type=String,Description="Orientation Bias prior for reverse compliment artifact">`
- `##FORMAT=<ID=OBPsnp,Number=1,Type=String,Description="Orientation Bias prior for real variant">`

dbSNP Annotation

In Germline, Tumor-Normal somatic, or Tumor-Only somatic modes, DRAGEN can look up variant calls in a dbSNP database and add annotations for any matches that it finds there. To enable the dbSNP database search, set the `--dbsnp` option to the full path to the dbSNP database VCF or `.vcf.gz` file, which must be sorted in reference order.

For each variant call in the output VCF, if the call matches a database entry for CHROM, POS, REF, and at least one ALT, then the rsID for the matching database entry is copied to the ID column for that call in the output VCF. In addition, DRAGEN adds a DB annotation to the INFO field for calls that are found in the database.

DRAGEN matches variant calls based on the name of the reference sequence/contig, but there is no additional way to assert that the reference used for constructing the dbSNP is the same as the reference used for alignment and variant calling. Make sure that the contigs in the selected annotation database match those in the alignment/variant calling reference.

Autogenerated MD5SUM for VCF Files

An MD5SUM file is generated automatically for VCF output files. This file is in the same output directory and has the same name as the VCF output file, but with an `.md5sum` extension appended. For example, `whole_genome_run_123.vcf.md5sum`. The MD5SUM files is a single-line text file that contains the md5sum of the VCF output file. This md5sum exactly matches the output of the Linux `md5sum` command.

Force Genotyping

DRAGEN supports force genotyping (ForceGT) for small variant calling. To use ForceGT, use the `--vc-forcegt-vcf` option with a list of small variants to force genotype. The input list of small variants can be a `*.vcf` or `*.vcf.gz` file.

The current limitations of ForceGT are as follows:

- ForceGT is supported for germline small variant calling in the V3 mode. The V1, V2, and V2+ modes are not supported.
- ForceGT is also supported for somatic small variant calling.
- ForceGT variants do not propagate through joint genotyping.

ForceGT Input

DRAGEN supports only a single ForceGT VCF input file, which must meet the following requirements:

- The input must be a valid VCF file according to version 4.2 of the VCF standard. For example, it must have at least eight tab-delimited columns, and records need to be sorted by reference contig and position.
- The header has to list the same contigs as the reference. All variants must refer to one of these contig names.
- Variants have to be normalized (parsimonious and left-aligned).
 - It must not contain any multinucleotide or complex variants (AT -> C). These are variants that require more than one substitution / insertion / deletion to go from REF allele to ALT allele and are ignored.
 - Deletions longer than 50bp are filtered out.
 - Variants will only be called once.
 - Duplicate entries are ignored.

The following nonnormalized variant will cause undefined behavior in DRAGEN:

instead of using...

- parsimonious: chrX 153592402 GC GCG

Use...

- parsimonious representation: chrX 153592403 C CG

ForceGT Operation and Expected Outcome

Force genotyping requires an input VCF and can be used with DRAGEN in VCF, GVCF or, VCF+GVCF mode. In all cases the output file(s) contains all regular calls and the forceGT variants, as follows:

- For a ForceGT call that was not called by the variant caller (not common), the call is tagged with FGT in the INFO field.
- For a germline ForceGT call that was also called by the variant caller and filter field is PASS, the call is tagged with NML;FGT in the INFO field (NML denotes normal). In somatic mode, the call is tagged with FGT;SOM.
- For a normal call (and PASS) by the variant caller, with no ForceGT call (normal), no extra tags are added (no NML tag, no FGT tag).

This scheme distinguishes among calls that are present due to FGT only, common in both ForceGT input and normal calling, and normal calls.

All the variants in the input ForceGT VCF are genotyped and present in the output file. The following table lists the reported GTs for the variants.

Condition	Reported GT
At a position with no coverage	./. or . .
At a position with coverage but no reads supporting ALT allele	0/0 or 0 0
At a position with coverage and reads supporting ALT allele	Dependent on pipeline (germline/somatic)

If DRAGEN calls a variant that is different from the one specified in the input ForceGT VCF, the output contains the following multiple entries at the same position:

- One entry for the default DRAGEN variant call
- One entry each for every variant call present in the input ForceGT VCF at that position.

```
chrX 100 G C [Default DRAGEN variant call]
chrX 100 G A [Variant in ForceGT vcf]
```

If a target BED file is provided along with the input ForceGT VCF, then the output file only contains ForceGT variants that overlap the BED file positions.

Variant Normalization

DRAGEN outputs variants in a VCF file following variant normalization as described here (https://genome.sph.umich.edu/wiki/Variant_Normalization). The normalization of a variant representation in VCF consists of two parts: parsimony and left alignment pertaining to the nature of a variant's length and position respectively.

- Parsimony means representing a variant in as few nucleotides as possible without reducing the length of any allele to 0.
- Left aligning a variant means shifting the start position of that variant to the left till it is no longer possible to do so.
- A variant is normalized if and only if it is parsimonious and left-aligned

Additional notes on variant representation in the DRAGEN VCF:

- Reference-trimming of alleles: A single padding reference base is used to represent insertions and deletions (i.e. the reference base preceding the insertion or deletion is included).

- Allele decomposition: multi-nucleotide polymorphisms (MNPs) are represented as separate, contiguous individual SNVs records in the VCF. If phasing can be determined, the FORMAT/GT is phased and the FORMAT/PS contains the coordinate position of the first variant in the set of phased variants. This determines which variant has occurred on the same haplotype.

In some cases, such as complex variants in repetitive regions, some variants cannot be normalized (ie, converted into a standard representation) or represented uniquely. To counteract this problem, when comparing two VCFs (eg, a DRAGEN VCF against a truth set VCF), it is recommended to use the `RTG vcfeval` tool which performs variant comparisons using a haplotype-aware approach.

Multi-allelic Variants and Overlapping Variants

A multiallelic site is a specific locus in a genome that contains three or more observed alleles, counting the reference as one, and therefore allowing for two or more variant alleles.

Multi-allelic calls are output in a single variant record in the VCF as follows:

SNP example:

```
chr1 2656216 . A T,C 107.65 PASS
AC=1,1;AF=0.500,0.500;AN=2;DP=12;FS=0.000;MQ=28.95;QD=8.97;SOR=3.056;FractionInInformativeReads=0.750 GT:AD:AF:DP:GQ:PL:GL:GP:PRI:SB:MB
1/2:0,5,4:0.556,0.444:9:15:177,144,46,122,0,72:-17.704,-14.420,-4.626,-
12.220,0.000,-7.244:1.076e+02,1.096e+02,1.465e+01,8.758e+01,1.520e-
01,4.082e+01:0.00,34.77,37.77,34.77,69.54,37.77:0,0,1,8:0,0,4,5
```

INDEL example:

Two indels are considered as multi-allelic if they share the same reference base preceding the indel.

```
chr1 7392258 . C CT,CTTT 234.76 PASS
AC=1,1;AF=0.500,0.500;AN=2;DP=44;FS=0.000;MQ=199.22;QD=5.34;SOR=2.226;FractionInInformativeReads=0.659 GT:AD:AF:DP:GQ:PL:GL:GP:PRI:SB:MB
1/2:0,15,14:0.517,0.483:29:50:245,256,55,190,0,55:-24.476,-25.634,-5.492,-
18.976,0.000,-5.500:2.348e+02,2.513e+02,5.292e+01,1.848e+02,4.401e-
05,5.300e+01:0.00,5.00,8.00,5.00,10.00,8.00:0,0,7,22:0,0,17,12
```

SNP and INDEL example:

A SNP and INDEL are considered multi-allelic if the SNP overlaps the reference base preceding the indel.

```
chr1 976778 . GC G,AC 70.96 PASS
AC=1,1;AF=0.500,0.500;AN=2;DP=4;FS=0.000;MQ=68.97;QD=17.74;SOR=3.258;FractionInInformativeReads=1.000 GT:AD:AF:DP:GQ:PL:GL:GP:PRI:SB:MB
```

```
1/2:0,2,2:0.500,0.500:4:32:137,70,68,74,0,73:-13.673,-6.979,-6.779,-
7.351,0.000,-7.325:7.096e+01,3.502e+01,3.602e+01,4.251e+01,2.828e-
03,4.525e+01:0.00,31.00,34.00,34.77,65.77,37.77:0,0,0,4:0,0,1,3
```

If a SNP overlaps an INDEL, but the SNP does not align with the reference base preceding the indel, the SNP and INDEL are represented as two different variant records, as shown in the example below:

```
chr1 1029628 . C CGT 49.88 PASS
AC=1;AF=0.500;AN=2;DP=37;FS=7.791;MQ=105.32;MQRankSum=-
1.315;QD=1.35;ReadPosRankSum=1.423;SOR=1.510;FractionInformativeReads=0.892;
R2_5P_bias=-19.742 GT:AD:AF:DP:GQ:PL:GL:GP:PRI:SB:MB:PS
0|1:17,16:0.485:33:48:81,0,50:-8.088,0.000,-5.000:4.988e+01,6.653e-
05,5.300e+01:0.00,31.00,34.00:10,7,5,11:11,6,9,7:1029628

chr1 1029629 . A G 50.00 PASS
AC=1;AF=0.500;AN=2;DP=37;FS=1.289;MQ=105.32;MQRankSum=-
0.659;QD=1.35;ReadPosRankSum=-
0.199;SOR=0.604;FractionInformativeReads=1.000;R2_5P_bias=-24.923
GT:AD:AF:DP:GQ:PL:GL:GP:PRI:SB:MB:PS 0|1:16,21:0.568:37:48:85,0,49:-
8.477,0.000,-4.934:5.000e+01,6.886e-
05,5.234e+01:0.00,34.77,37.77:9,7,10,11:10,6,13,8:1029628
```

1. RTG vcfeval has been adopted as the standard VCF comparison tool by GA4GH and PrecisionFDA (<https://www.biorxiv.org/content/biorxiv/early/2018/02/23/270157.full.pdf>).

DRAGEN-ML

DRAGEN employs machine learning-based variant recalibration (DRAGEN-ML) for germline SNV VC. Variant calling accuracy is improved using powerful and efficient machine learning techniques that augment the variant caller, by exploiting more of the available read and context information that does not easily integrate into the Bayesian processing used by the haplotype variant caller. A supervised machine learning method was developed using truth from the PrecisionFDA v4.2.1 sets to build a model that processes read and other contextual evidence to remove false positives, recover false negatives, and reduce zygosity errors for both SNVs and INDELS.

Setup

Additional setup is not required. ML model files for the hg38 and hg19 human references are packaged with the DRAGEN installer.

- After the installation, the files will be present in the `/opt/edico/resources/ml_model/<ref>` folder.
- DRAGEN-ML is enabled as needed, when running the germline SNV VC.

- DRAGEN detects the reference used for analysis, and will use the correct model files. If hg38 or hg19 reference types are not detected, then ML recalibration will automatically be disabled and SNV VC falls back to legacy operation.

Command-line Options

Example DRAGEN CMD line options:

```
--vc-ml-dir=/path/to/package/directory --vc-ml-enable-recalibration=true
```

Where /path/to/package/directory contains the extracted support files from the package for DRAGEN-ML

Inputs

Since the machine learning model extracts information from the read pile-up, DRAGEN-ML requires a run with BAM or FASTQ input. DRAGEN-ML runs concurrently with DRAGEN SNV VC. DRAGEN-ML can be applied to WGS or WES samples. Re-calibration of existing VCF files is not supported.

Outputs

DRAGEN-ML recalibrates all quality scores, changing the values of the QUAL and GQ fields in the output VCF/GVCF.

- DRAGEN-ML also updates PL and GP in the output VCF/GVCF.
- The genotypes (GT field) of some variants may be changed by ML eg, 0/1 to 1/1 or vice versa.
- DRAGEN-ML PHRED scores are limited to a maximum value of around 60-70. Therefore, the QUAL filtering threshold is set to 3 when DRAGEN-ML is enabled, compared to 10 for DRAGEN-VC when DRAGEN-ML is disabled.

The following variant types are re-calibrated:

- Biallelic and multiallelic variants
- Autosomes and sex chromosomes, including haploid positions
- Force GT calls
- Non primary contigs

Accuracy Improvements

DRAGEN-ML typically removes 30-50% of SNP FPs, with smaller gains on INDELS. FN counts are reduced by 10% or more. The output QUAL/GQ of DRAGEN-ML is empirically more accurately calibrated than DRAGEN SNV VC without ML. There are significant gains in accuracy statistics across the entire genome with ML enabled. Note that a small number of variant calls may have degraded accuracy with ML enabled compared to VC without ML.

Run time

DRAGEN-ML adds about 10% to the run time compared to runs without ML.

VC Evidence BAM

The DRAGEN small variant caller is a haplotype-based caller which performs local assembly of all reads in an active region into a de Bruijn graph (DBG). The assembly process uses all the read bases including the soft-clip bases of reads. The soft-clip bases provide evidence for the presence of variants, specifically longer insertions and deletions which are not present in the read cigar and hence cannot be directly viewed in IGV.

The assembly and realignment step (using pair-HMM) performed by variant caller aims to correct mapping errors made by the original aligner and improves the overall variant caller accuracy. Using the evidence BAM, we can view how the variant caller sees the read evidence and how the reads have been realigned making it a very useful debugging tool.

By default, the evidence BAM contains only a subset of regions processed by the small variant caller. Only regions which have candidate indel variants and some percentage of soft-clip reads in the pile up are realigned and output in the evidence BAM. This is done to reduce the run time overhead needed to generate the evidence BAM.

Outputs

A BAM file with the suffix `_evidence.bam` and the corresponding index file. The evidence BAM can be enabled along with the regular BAM output from the Map-Align step. When multiple BAM are passed as inputs to the variant caller, for eg, in Tumor-Normal calling, then they will be combined in the evidence BAM output and tagged with appropriate read groups.

Features

The evidence BAM consists of realigned reads, badly mated reads and reads that are disqualified by the variant caller based on the read likelihood scores.

Disqualified and Badly Mated reads

Reads that are badly-mated (when the read and its mate are mapped to different chromosomes) are tagged with a BM tag (integer) and reads that are disqualified (based on read likelihoods) are tagged with the DQ tag (integer). These reads are filtered out by the genotyper in the variant caller. The alignment score tag AS is forced to 0 for such reads in the evidence BAM and hence, they can be filtered from the IGV pile up by setting the minimum AS score to be 1 instead of 0.

Graph Haplotypes

When enabling graph haplotypes output using `--vc-evidence-bam-output-haplotypes`, all the haplotypes constructed by the de Bruijn graph are output in the evidence BAM as single reads covering the entire active region. The reads and haplotypes are tagged with different read groups which makes it

easily distinguishable in IGV. In IGV, we can use “Color Alignments By” or “Group Alignments By” > read group to separate out the reads from the haplotypes. The haplotypes are tagged with read group EvidenceHaplotype and the reads are part of the EvidenceRead_Normal/Tumor read group.

The haplotypes are named as Haplotype 1, Haplotype 2 and so on and have an additional ‘HC’ tag (integer). The realigned reads also have an HC tag which encodes which haplotype best matches the read based on the likelihood calculation. Only reads which are supported by a single unique haplotype have the HC tag, reads which match more than one haplotype well do not have an HC tag. The use of this tag is primarily intended to enable highlighting of reads in IGV. Go to "Color Alignments By > Tag" and enter "HC" to view which reads are uniquely supported by a certain graph haplotypes.

Command Line Arguments

Name	Description	Default Value
<code>vc-output-evidence-bam</code>	Enable evidence BAM output	False
<code>vc-evidence-bam-output-haplotypes</code>	Output graph haplotypes in evidence BAM	False
<code>vc-evidence-bam-clipped-read-threshold</code>	Percentage of clipped reads in active region to enable evidence BAM output for that region	10%
<code>vc-evidence-bam-force-output</code>	Force evidence BAM output for all active regions	False

High Sensitivity Mode

The default mode of the DRAGEN small variant caller has been optimized to produce the lowest FP+FN count and to maximize the f-measure score for both SNPs and INDELs. For use cases where it is important to have lower FN calls, even at the expense of extra FP calls, a high sensitivity mode is available. There are two main changes when high sensitivity mode is enabled:

- Detection of variants in regions where all reads have `MAPQ=0` Unlike in default mode, when high sensitivity mode is enabled, regions that only have `MAPQ=0` reads are processed in the same way as other regions to call variants. Since the read alignments in these regions are ambiguous and to allow quick identification of such calls, the variants are tagged with `INFO/EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION`.
- Detection of low AF calls The default DRAGEN ML model has been trained to identify germline calls with typical AFs of 0%, 50% or 100%. To improve sensitivity of low AF calls, a new ML model trained using read and context evidence from both germline and low AF calls is used. This allows the model to identify variants down to approximately 5% AF. The hard filter QUAL threshold for both SNPs and INDELs is lowered to 0.4 in this mode to allow low AF calls to be set as PASS in the FILTER field.

Command line options

--vc-enable-high-sensitivity-mode	Set to true to enable DRAGEN small variant caller high sensitivity mode.
-----------------------------------	--

VCF Imputation

The VCF imputation tool can infer multi-allelic SNP and Indel variants from low-coverage sequencing samples by packaging the GLIMPSE software (2020, Olivier Delaneau & Simone Rubinacci). The DRAGEN implementation of the GLIMPSE software allows for scalability of variant imputation by adding the following features:

- End-to-end pipeline, where the 3 phases of the GLIMPSE software (Chunk, Phase, and Ligate) get executed by a single command, on one chromosome or multiple chromosomes
- Software supported acceleration

The DRAGEN VCF imputation tool infers variants on autosomes and chromosome X of haploid and diploid species.

Upon completion, the tool generates imputed variants based on a reference panel, a genetic map, and input samples provided. The DRAGEN Bio-It Platform supports VCF imputation on human data and provides a reference panel and a genetic map for the hg38 reference build accessible on the [Illumina DRAGEN Bio-IT Platform Product Files](#) page.

For data other than human data (reference build hg38) the user needs to provide its own reference panel and genetic map. A custom reference panel can be built with the DRAGEN Population Haplotyping tool.

- i**
- The output is in biallelic format and requires post-collapsing for downstream analyses, even though the imputation tool can impute multi-allelic positions.
 - The VCF imputation tool only supports input sample data generated with the DRAGEN Bio-It Platform.

The following is an example of commands to impute vcf on a single chromosome:

```
dragen
--enable-imputation true
--imputation-ref-panel-dir <REF_PANEL_DIR>
--imputation-ref-panel-prefix <IRPv2.0>
--imputation-chunk-input-region <chr22>
--imputation-phase-input-list <VCF_to_be_imputed.txt>
--imputation-genome-map-dir <MAP_DIR>
--output-directory <OUT_DIR>
--output-file-prefix <OUT_PREFIX>
```

The following is an example of commands to impute vcf on chromosome X:

```
dragen
--enable-imputation true
--imputation-ref-panel-dir <REF_PANEL_DIR>
--imputation-ref-panel-prefix <IRPv2.0>
--imputation-chunk-input-region <chrX>
--imputation-phase-input-list <VCF_to_be_imputed.txt>
--imputation-genome-map-dir <MAP_DIR>
--imputation-phase-sample-type-list <path to sample type file>
--output-directory <OUT_DIR>
--output-file-prefix <OUT_PREFIX>
```

Inputs

Sample Input

The imputation tool infers multi-allelic SNP and INDEL variants from low-coverage sequencing samples that are provided by the user. To maximize the accuracy of the imputed variant per sample, the tool leverages the information from all provided samples.

The sample(s) to be imputed must have the following format:

- VCF, multisample VCF, BCF or multisample BCF (zip or unzipped). gVCF is not supported
- Must contain GL (Genotype Likelihoods) or PL (Phred-scaled genotype likelihoods) information

To achieve more accurate results, it is recommended to use input VCF generated with the force genotyping capability of the DRAGEN Bio-It Platform so that it contains all the positions that are present in the reference panel. A file to be used as input of the force genotyping run of the DRAGEN variant caller, with all sites present in the IRP reference panel (built from human reference genome hg38) is provided in the Imputation files accessible in the [Illumina DRAGEN Bio-IT Platform Support Site](#) page. When running the force genotype option (of DRAGEN variant caller) for imputation, it is recommended to disable the machine learning tool `--vc-ml-enable-recalibration=false`.

Reference Panel

A per-chromosome reference panel in VCF or BCF format that lists all the imputation positions in the targeted regions along with the corresponding haplotypes must be provided. A reference panel (with prefix IRPv{x}) is available in the imputation files accessible in the [Illumina DRAGEN Bio-IT Platform Support Site](#) page. IRPv2.0 is a multi-allelic SNP, INDELS reference panel containing the 3202 samples from the 1000 Genomes Project, which have been variant called using DRAGEN 4.0 against hg38.

 IRPv1.x does not support chrX

IRPv2.x supports chrX

ChrY and chrM are not supported

A custom reference panel can be built with the DRAGEN Population Haplotyping tool. When providing a custom reference panel ensure the chromosome of mixed ploidy chromosome is divided into the PAR and non-PAR regions that exist, and the basename matches the subregions names defined in the JSON config file. The format should be <PREFIX>.basename. Examples: IRPv2.0.chrX.par1, IRPv2.0.chrX.par2, and IRPv2.0.chrX.nonpar.

Genetic Map

A genetic map per chromosome is required to obtain the imputed variants. You can use your own genetic map computed from the recombination rate of the species and its reference genome, or use the genetic map corresponding to the human hg38 reference genome available in the Imputation files accessible in the [Illumina DRAGEN Bio-IT Platform Support Site](#) page. DRAGEN does not generate custom genetic map files.

The genetic map should follow the format:

- <chromosome name>.gmap.gz
- 3 columns: position, chromosome number, distance (cM)
- compliant with the reference genome used to generate the sample input

JSON config file

This config file allows the proper handling of haploid/diploid chromosomes. This file is present in the same directory of the input reference panel with PREFIX and is available in the [Illumina DRAGEN Bio-IT Platform Support Site](#) page. It must follow the naming convention: \${DIR}/\${PREFIX}.config.json. When the config file is not present in the directory, the tool assumes that the imputation is done on all diploid chromosomes.

In the IRP reference panel folder available on DRAGEN support page, the JSON config file corresponds to human data. The user can edit this file if imputation is done on another species.

Example of JSON config file

For imputing VCF on human data with typename “M” for Male and “F” for Female (“M” and “F” are the values used in the sample type file):

```
{
  "regions": {
    "chrX" : [ "chrX_par1", "chrX_nonpar", "chrX_par2" ]
  },
}
```

```
"ploidy" : {
  "chrX_nonpar" : { "M": 1, "F": 2 },
  "default" : { "M": 2, "F": 2 }
}
```

Instructions to make a custom JSON configuration file:

The JSON config file is made of two fields as defined in the table below

Fields	Required/Optional	Purpose	Type
regions	Required only when a chromosome of mixed ploidy is present in the Reference Panel folder	Define contig name and subregion name of mixed ploidy chromosome	Dictionary in the form: contigname_of_mixed_ploidy : [contigname_of_mixed_ploidy"_par1", contigname_of_mixed_ploidy"_par2", contigname_of_mixed_ploidy"_nonpar1", contig_name_of_mixed_ploidy"_nonpar2" ...]
ploidy	<ul style="list-style-type: none"> "default" is a required name contigname_of_mixed_ploidy_"nonpar" is required only when a chromosome of mixed ploidy is present in the Reference Panel folder 	<p>Define:</p> <ul style="list-style-type: none"> ploidy behavior when different from "default" default ploidy behavior 	Dictionary in the form: contigname_of_mixed_ploidy_"nonpar": { typename1 : 1, typename2 : 2} "default" : { "typename1": 2, "typename 2": 2} typename is used in the Sample Type file input

i | The subregion names must match the genetic map name.

For Example, if `chrX_nonpar` is defined in the `region` field of the JSON config file, then the genetic map corresponding to chromosome X non PAR region must be named `chrX_nonpar.gmap.gz`.

Sample type file

The sample type file is required when haplotyping is performed on non-PAR regions of mixed ploidy chromosomes to define the typename of each sample.

The sample type file is a txt file with the following format

- 2 columns, tabs or space delimited
- First column: list of all sample names present in the input sample
- Second column: typename value for each sample. This typename value should match the typenames used in the JSON config file.

Outputs

The VCF imputation tool generates a couple outputs:

- Imputed variant file with concatenated imputed variants: one VCF or msVCF file which contains all the specified regions/chromosomes with name `<prefix>.impute.vcf.gz`
- Intermediate files:
 - Chunk regions to be passed along to the internal phase step with the name `<prefix>.impute.chunk.out.txt`.
 - Imputed variants per chunks identified: VCF or msVCF file based on the input sample format with the name `<prefix>_chr_start-end.impute.phase.vcf.gz`.
 - A text file with the path to all the `<prefix>_chr_start-end.impute.phase.vcf.gz` files generated with the name `<prefix>.impute.phase.out.txt`.

i | While the imputation tool can impute multi-allelic positions, the output is in biallelic format and requires collapsing for downstream analyses.

Command-Line Options

Option	Type	Required	Description
<code>--enable-imputation</code>	NA	Yes	Set to true to enable vcf imputation pipeline

Option	Type	Required	Description
--imputation-ref-panel-dir	STRING	Yes	Directory containing per-chromosome reference panel VCF and optionally the JSON config file
--imputation-ref-panel-prefix	STRING	Yes	Prefix for reference panel files and the JSON config file
--imputation-genome-map-dir	STRING	Yes	Directory containing per-chromosome genome map files
--imputation-chunk-input-region	STRING	Yes for single region	Target region, usually a full chromosome (eg chr20:1000000-2000000 or chr20).
--imputation-chunk-input-region-list	STRING	Yes for list of regions	Text file listing chromosomes or regions to be processed, one chromosome/region per line.
--imputation-phase-input-list	STRING	Yes for multiple VCF files	Text file listing sample input in VCF/BCF format, one input file per line
--imputation-phase-sample-type	STRING	Yes, when imputing on a non PAR region of mixed ploidy chromosome AND a single VCF file.	Define typename of the VCF file imputed. The typename must match one of the two typenames defined in the JSON config file
--imputation-phase-sample-type-list	STRING	Yes, when imputing on a non PAR region of mixed ploidy chromosome AND a list of VCF files.	Path to the Sample Type file

Option	Type	Required	Description
--output-directory	STRING	Yes	Output directory
--output-file-prefix	STRING	Yes	Output files prefix
--imputation-phase-threads	INT	No	Specify the number of threads to use. Default is the number of system threads
--imputation-phase-filter-input-sample-in-ref	NA	No	Default is <code>true</code> . If sample ID matches between reference panel and sample input, then the corresponding samples are ignored from the reference panel to avoid imputation against itself. To be turned to <code>false</code> if all samples from the reference panel should be kept regardless of their presence in the sample input.
--imputation-phase-impute-reference-only-variants	STRING	No	Default is <code>false</code> . If set to <code>true</code> allows imputation at variants only present in the reference panel. The use of this option is intended only to allow imputation at sporadic missing variants. If the number of missing variants is non-sporadic, please re-run the genotype likelihood computation at all reference variants and avoid using this option, because data from the reads should be used.

Option	Type	Required	Description
--impute-phase-input-independently	STRING	No	Default is false If set to true allows to treat each sample input independently without using them in the reference panel calculation

The end-to-end implementation of the GLIMPSE software are set with the following parameters:

- window_size = 2 Mb
- buffer_size = 200 kb

Copy Number Variant Calling

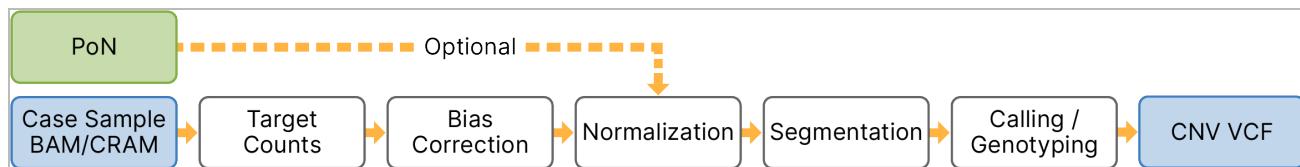
The DRAGEN Copy Number Variant (CNV) Pipeline can call CNV events using next-generation sequencing (NGS) data. This pipeline supports multiple applications in a single interface via the DRAGEN Host Software, including processing of whole-genome sequencing (WGS) data and whole-exome sequencing (WES) data for germline analysis.

The DRAGEN CNV pipeline supports two normalization modes of operation. The two modes apply different normalization techniques to handle biases that differ based on the application, for example, WGS versus WES. While the default option settings attempt to provide the best trade-off in terms of speed and accuracy, a specific workflow may require more finely tuned option settings.

CNV Workflow

The DRAGEN CNV pipeline follows the workflow shown in the following figure.

Figure 2 DRAGEN CNV Pipeline Workflow



The DRAGEN CNV Pipeline uses many aspects of the DRAGEN platform available in other pipelines, such as hardware acceleration and efficient I/O processing. To enable CNV processing in the DRAGEN Host Software, set the `--enable-cnv` command line option to true.

The CNV pipeline has the following processing modules:

- Target Counts—Binning of the read counts and other signals from alignments.
- Bias Correction—Correction of intrinsic system biases.

- Normalization—Detection of normal ploidy levels and normalization of the case sample.
- Segmentation—Breakpoint detection via segmentation of the normalized signal.
- Calling / Genotyping—Thresholding, scoring, qualifying, and filtering of putative events as copy number variants.

The normalization module can optionally take in a panel of normals (PoN), which is used when a cohort or population samples are readily available. All other modules are shared between the different CNV algorithms.

Signal Flow Analysis

The following figures show a high-level overview of the steps in the DRAGEN CNV Pipeline as the signal traverses through the various stages. These figures are examples and are not identical to the plots that are generated from the DRAGEN CNV Pipeline.

The first step in the DRAGEN CNV Pipeline is the target counts stage. The target counts stage extracts signals such as read count and improper pairs and puts them into target intervals.

Figure 3 Read Count Signal

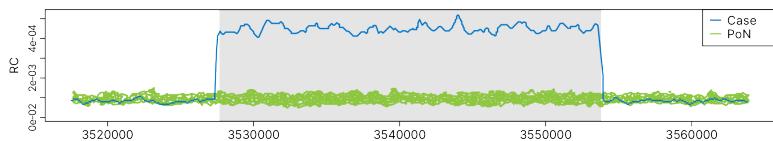
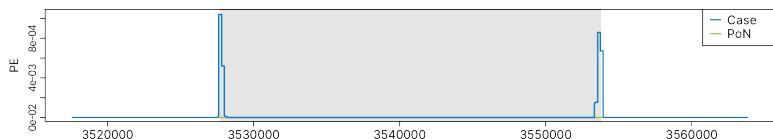


Figure 4 Improper Pairs Signal

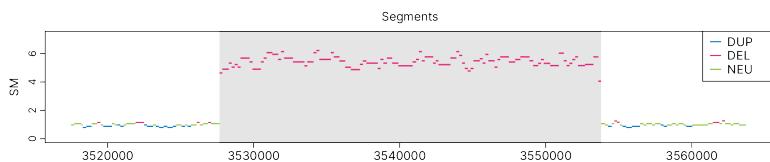
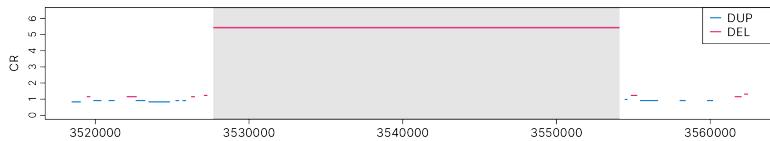


Next, the case sample is normalized against the panel of normals or against the estimated normal ploidy level. Any other biases are subtracted out of the signal to amplify any event level signals.

Figure 5 Pre/Post Tangent Normalization



The normalized signal is then segmented using one of the available segmentation algorithms. Events are then called from the segments.

Figure 6 Segments**Figure 7 Called Events**

The events are then scored and emitted in the output VCF.

CNV Pipeline Options

The following are the top-level options that are shared with the DRAGEN Host Software to control the CNV pipeline. You can input a BAM or CRAM file into the CNV pipeline. If you are using the DRAGEN mapper and aligner, you can use FASTQ files.

Option	Description
<code>--bam-input</code>	The BAM file to be processed.
<code>--cram-input</code>	The CRAM file to be processed.
<code>--enable-cnv</code>	Enable or disable CNV processing. Set to true to enable CNV processing.
<code>--enable-map-align</code>	Enables the mapper and aligner module. The default is true, so all input reads are remapped and aligned unless this option is set to false.
<code>--fastq-file1</code>	FASTQ file or files to be processed.
<code>--fastq-file2</code>	
<code>--output-directory</code>	Output directory where all results are stored.
<code>--output-file-prefix</code>	Output file prefix that will be prepended to all result file names.
<code>--ref-dir</code>	The DRAGEN reference genome hashtable directory.

CNV Pipeline Input

The DRAGEN CNV pipeline supports multiple input formats. The most common format is an already mapped and aligned BAM or CRAM file. If you have data that has not yet been mapped and aligned, see [Generate an Alignment File on page 180](#).

To run the DRAGEN CNV pipeline directly with FASTQ input without generating a BAM or CRAM file, see [Streaming Alignments on page 181](#) for instructions on streaming alignment records directly from the DRAGEN map/align stage.

Reference Hashtable

For the DRAGEN CNV pipeline, the hashtable must be generated with the `--enable-cnv` option set to true, in addition to any other options required by other pipelines. When `--enable-cnv` is true, DRAGEN generates an additional k-mer uniqueness map that the CNV algorithm uses to counteract mapability biases. You only need to generate the k-mer uniqueness map file one time per reference hashtable. The generation takes about 1.5 hours per whole human genome.

The reference hashtable is a pregenerated binary representation of the reference genome. For information on generating a hashtable, see [Prepare a Reference Genome on page 11](#).

The following example command generates a hashtable.

```
dragen \
--build-hash-table true \
--ht-reference <FASTA> \
--output-directory <OUTPUT> \
--enable-cnv true \
<OTHER HASHTABLE OPTIONS> \
```

Generate an Alignment File

The following command-line examples show how to run the DRAGEN map/align pipeline depending on your input type. The map/align pipeline generates an alignment file in the form of a BAM or CRAM file that can then be used in the pipeline.

You need to generate alignment files for all samples that have not already been mapped and aligned. Each sample must have a unique sample identifier. Use the `--RGSM` option to specify the identifier. For BAM and CRAM input files, the sample identifier is taken from the file, so the `--RGSM` option is not required.

The following example command maps and aligns a FASTQ file:

```
dragen \
-r <HASHTABLE> \
-1 <FASTQ1> \
-2 <FASTQ2> \
--RGSM <SAMPLE> \
--RGID <RGID> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align true
```

The following example command maps and aligns an existing BAM file:

```
dragen \
-r <HASHTABLE> \
--bam-input <BAM> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align true
```

The following example command maps and aligns an existing CRAM file:

```
dragen \
-r <HASHTABLE> \
--cram-input <CRAM> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align true
```

Streaming Alignments

DRAGEN can map and align FASTQ samples, and then directly stream them to downstream callers, such as the CNV Caller and the Haplotype Variant Caller. You can use this process to skip generation of a BAM or CRAM file, which bypasses the need to store additional files.

To stream alignments directly to the DRAGEN CNV pipeline, run the FASTQ sample through a regular DRAGEN map/align workflow, and then provide additional arguments to enable CNV. The following example command line maps and aligns a FASTQ file, and then sends the file to the Germline CNV WGS pipeline.

```
dragen \
-r <HASHTABLE> \
-1 <FASTQ1> \
-2 <FASTQ2> \
```

```
--RGSM <SAMPLE> \
--RGID <RGID> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align true \
--enable-cnv true \
--cnv-enable-self-normalization true
```

For information on running CNV concurrently with the Haplotype Variant Caller, see [Concurrent CNV and Small Variant Calling on page 204](#).

Target Counts

The target counts stage is the first processing stage for the DRAGEN CNV pipeline. This stage bins the alignments into intervals. The primary analysis format for CNV processing is the target counts file, which contains the feature signals that are extracted from the alignments to be used in downstream processing. The binning strategy, interval sizes, and their boundaries are controlled by the target counts generation options, and the normalization technique used.

When working with whole genome sequence data, the intervals are autogenerated from the reference hashtable. Only the primary contigs from the reference hashtable are considered for binning. You can specify additional contigs to bypass with the `--cnv-skip-contig-list` option.

With whole exome sequence data, DRAGEN uses the target BED file supplied with the `--cnv-target-bed` option to determine the intervals for analysis.

The target counts stage generates a `.target.counts.gz` file. You can use the file later in place of any BAM or CRAM by specifying the file with the `--cnv-input` option for the normalization stage. The `.target.counts.gz` file is an intermediate file for the DRAGEN CNV pipeline and should not be modified.

The target counts stage generates a `.target.counts.gz` file. You can use the file later in place of any BAM or CRAM by specifying the file with the `--cnv-input` option for the normalization stage. The `.target.counts.gz` file is an intermediate file for the DRAGEN CNV pipeline and should not be modified.

The `.target.counts.gz` file is a tab-delimited compressed text file with the following columns:

- Contig identifier
- Start position
- End position
- Target interval name
- Count of alignments in this interval
- Count of improperly paired alignments in this interval

Header lines that start with # are also included. It contains the DRAGEN version and command line that was used to generate the file, as well as other meta information.

An example of a *.target.counts.gz file is shown below.

```
#TARGET COUNTS FILE

##DRAGENVersion=<VERSION_INFO>

##DRAGENCommandLine=<CommandLineOptions>

...

contig    start    stop    name          SampleName    improper_pairs
1         565480   565959  target-wgs-1-565480 7       6
1         566837   567182  target-wgs-1-566837 9       0
1         713984   714455  target-wgs-1-713984 34      4
1         721116   721593  target-wgs-1-721116 47      1
1         724219   724547  target-wgs-1-724219 24      21
1         725166   725544  target-wgs-1-725166 43      12
1         726381   726817  target-wgs-1-726381 47      14
1         753243   753655  target-wgs-1-753243 31      2
1         754322   754594  target-wgs-1-754322 27      0
1         754594   755052  target-wgs-1-754594 41      0
```

Whole Genome

If the samples are whole genome, then the effective target intervals width is specified with the `--cnv-interval-width` option. The higher the coverage of a sample, the higher the resolution that can be detected. This option is important when running with a panel of normals because all samples must have matching intervals. For self-normalization, the effective width might be larger than the specified value.

The default value for WGS is 1000 bp with a sample coverage of $\geq 30x$.

WGS Coverage per Sample	Recommended Resolution* (bp)
5	10000
10	5000
≥30	1000

*Using a `cnv-interval-width` of ≤250 bp for WGS analysis can drastically increase run time

The intervals are autogenerated for every primary contig in the reference. DRAGEN only supports references that have the USCS or GRC convention. For example, `chr1`, `chr2`, `chr3`, ..., `chrX`, `chrY` or `1`, `2`, `3`, ..., `X`, `Y`. To specify a list of contigs to skip, use the `--cnv-skip-contig-list` option. This option takes comma-separated list of contig identifiers. The contig identifiers must match the reference hashtable that you are using. By default, only the mitochondrial chromosomes are skipped. Nonprimary contigs are never processed.

For example, to skip chromosome M, X, and Y, use the following option:

```
--cnv-skip-contig-list "chrM,chrX,chrY"
```

Whole Exome

If the samples are whole exome samples, supply a target BED file with the `--cnv-target-bed $TARGET_BED` option.

The intervals in the target BED file indicate regions where alignments are expected based on the target capture kit. The BED file intervals are further split into intervals of smaller size, depending on the value of `cnv-interval-width`.

To use a standard BED file, make sure that there is no header present in the file. In this case, all columns after the third column are ignored, similar to the operation of DRAGEN Variant Caller.

Target Counts Options

The following options control the generation of target counts.

Option	Description
<code>--cnv-counts-method</code>	Specifies the counting method for an alignment to be counted in a target bin. Values are midpoint, start, or overlap. The default value is overlap when using the panel of normals approach, which means if an alignment overlaps any part of the target bin, the alignment is counted for that bin. In the self-normalization mode, the default counting method is start.

Option	Description
--cnv-min-mapq	Specifies the minimum MAPQ for an alignment to be counted during target counts generation. The default value is 3 for self-normalization and 20 otherwise. When generating counts for panel of normals, all MAPQ0 alignments are counted.
--cnv-target-bed	Specifies a properly formatted BED file that indicates the target intervals to sample coverage over. For use in WES analysis.
--cnv-interval-width	Specifies the width of the sampling interval for CNV processing. This option controls the effective window size. The default is 1000 for WGS analysis and 500 for WES analysis.
--cnv-skip-contig-list	Specifies a comma-separated list of contig identifiers to skip when generating intervals for WGS analysis. The default contigs that are skipped, if not specified, are chrM, MT, m, chrm.
--cnv-count-duplicate-alignments	Keep duplicate marked alignments during target counts if option is set to true.

Count Duplicate Alignments

PCR duplicates are often considered as noise in coverage depth information. DRAGEN CNV has an option to include/exclude duplicate marked alignments: `--cnv-count-duplicate-alignments` when counting alignments. This relies on the alignments having the duplicate-marked bit (0x400) in the SAM flag set correctly. There is a limitation of use cases based on the input file type. If the input file is not a duplicate marked BAM, then the read duplication information is not present, and all reads will be used during the target counts process. To exclude duplicate alignments from FASTQ or unaligned BAM inputs, run DRAGEN map/aligner with duplicate marking enabled, and then run DRAGEN CNV with the resulting output BAM. For WES or targeted panel application, it is recommended to have the same behavior and settings applied during Panel of Normals generation and how the case sample will be processed. The option `--cnv-count-duplicate-alignments` allows you to control whether duplicate alignments are retained or filtered out, but relies on the duplicate marked flag. The following block diagram shows various scenarios where this option is applicable:

Input format	enable-map-align	Duplicate marked alignments considered in Target Counts?
FASTQ	TRUE	Always included
BAM	TRUE	Always included
BAM	FALSE	Configurable with cnv-count-duplicate-alignments

Target Counts Dropout Regions

In the WGS case where a BED file is not specified for a given reference, the same intervals should be generated each time. The intervals created take into account the mappability of the reference genome using a k-mer uniqueness map created during hashtable generation.

Due to ambiguity that may arise from non-unique genomic loci, only regions corresponding to unique k-mers are considered. A position in the reference genome is marked as unique if the k-mer starting at that position does not show up anywhere else in the reference genome (or non-unique, otherwise). If the k-mer contains any bases other than A, C, T, or G, it is marked as non-unique.

For WGS samples, and in absence of a `cnv-target-bed` file, the target intervals are auto generated based on the pre-computed k-mer-uniqueness map for a given input reference hashtable, and the `cnv-interval-width` option which defaults to 1000bp. The `cnv-interval-width` option determines the minimum number of unique k-mer positions required in the interval. There is an upper bound to the length of the interval: when the length of the interval is greater than double the size of `cnv-interval-width`, without reaching the required count of unique k-mer positions, the interval is discarded and the process starts again at the next genomic position. Regions that are discarded are denoted as dropout regions with the `NON_KMER_UNIQUE` exclusion reason in the `*.cnv.excluded_intervals.bed.gz` file.

A dropout region is a complex region that does not count alignments and results in an interval missing from the analysis. Dropout regions include centromeres, telomeres, and low complexity regions. If there is sufficient signal in the flanking regions, an event can still span these dropout regions, even if alignment counting does not occur in the regions. The event is handled by the segmentation stage.

GC Bias Correction

GC Biases measure the relationship between GC content and read coverage across a genome. Biases can occur in library prep, capture kits, sequencing system differences, and mapping. Biases can result in difficulties calling CNV events. The DRAGEN GC bias correction module attempts to correct these biases.

The GC bias correction module immediately follows the target counts stage and operates on the `*.target.counts.gz` file. GC bias correction generates a GC bias corrected version of the file, which has a `*.target.counts.gc-corrected.gz` extension in the file name. The GC bias corrected versions are recommended for any downstream processing when working with WGS data. For WES, if there are enough target regions, then the GC bias corrected counts can also be used.

Typical capture kits have over 200,000 targets spanning the regions of interest. If your BED file has fewer than 200,000 targets, or if the target regions are localized to a specific region in the genome (such that GC bias statistics might be skewed), then GC bias correction should be disabled.

The following options control the GC bias correction module.

Option	Description
--cnv-enable-gcbias-correction	Enable or disable GC bias correction when generating target counts. The default is true.
--cnv-enable-gcbias-smoothing	Enable or disable smoothing the GC bias correction across adjacent GC bins with an exponential kernel. The default is true.
--cnv-num-gc-bins	Specifies the number of bins for GC bias correction. Each bin represents the GC content percentage. Allowed values are 10, 20, 25, 50, or 100. The default is 25.

Normalization

The DRAGEN CNV pipeline supports two normalization algorithms:

- Self-Normalization—Estimates the autosomal diploid level from the sample under analysis to determine the baseline level to normalize by. Sex chromosomes and PAR regions are handled based on the sample sex.
- Panel of Normals—A reference-based normalization algorithm that uses additional matched normal samples to determine a baseline level from which to call CNV events. The matched normal samples means it has undergone the same library prep and sequencing workflow as the case sample.

Which algorithm to use depends on the available data and the application. Use the following guidelines to select the mode of normalization.

Self-Normalization

- Whole genome sequencing
- Single sample analysis
- Additional matched samples are not readily available
- Simpler workflow via a single invocation
- Only references with chr1, chr2, chr3, ..., chrX, chrY or 1, 2, 3, ..., X, Y naming conventions are supported

Panel of Normals

- Whole genome sequencing
- Whole exome sequencing
- Targeted panels, including somatic panels
- Additional matched samples are available
- Nonhuman samples

Self Normalization

The DRAGEN CNV pipeline provides the self-normalization mode that does not require a reference sample or a panel of normals. To enable this mode, set `--cnv-enable-self-normalization` to true. Self-normalization mode bypasses the need to run two stages and can save time. It uses the statistics within the case sample to determine the baseline from which to make a call.

Because self-normalization uses the statistics within the case sample, this mode is not recommended for WES or targeted sequencing analysis due to the potential for insufficient data.

The self-normalization mode is the recommended approach for whole-genome sequencing single sample processing. The pipeline continues through to the segmentation and calling stage to produce the final called events.

```
dragen \
-r <HASHTABLE> \
--bam-input <BAM> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
--enable-cnv true \
--cnv-enable-self-normalization true
```

If you are running from a FASTQ sample, then the default mode of operation is self-normalization.

When operating in self-normalization mode, the `--cnv-interval-width` option used during the target counts stage becomes the effective interval width based on the number of unique k-mer positions. You typically do not have to modify this option.

Self-normalization autogenerates the target intervals for use during analysis based on the reference genome and is only compatible with standard human references. Nonstandard human references require a BED file to process and the panel of normals approach.

Panel of Normals

The panel of normals mode uses a set of matched normal samples to determine the baseline level from which to call CNV events. These matched normal samples should be derived from the same library prep and sequencing workflow that was used for the case sample. This allows the algorithm to subtract system level biases that are not sample specific. The generation of the target counts for the normal samples should also have identical command line options with the case sample under analysis.

In this mode, the DRAGEN CNV Pipeline is broken down into two distinct stages. The target counts stage is performed on each sample, case, and normals, to bin the alignments. The normalization and call detection stage is then performed with the case sample against the panel of normals to determine the events.

Target Counts Stage

Perform target counts for all your samples, whether the samples are to be used as references or are the case samples under analysis. The case samples and all samples to be used as a panel of normals sample must have identical intervals and therefore should be generated with identical settings. The target counts stage also performs GC Bias correction. GC Bias correction is enabled by default.

The following examples are for WES processing, and is the case when a panel of normals is required.

The following is an example command for processing a BAM file.

```
dragen \
-r <HASHTABLE> \
--bam-input <BAM> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
--enable-cnv true \
--cnv-target-bed <BED> \
```

The following is an example command for processing a CRAM file.

```
dragen \
-r <HASHTABLE> \
--cram-input <CRAM> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
--enable-cnv true \
--cnv-target-bed <BED> \
```

Normalization and Call Detection Stage

The next step in the CNV pipeline when using a panel of normals is to perform the normalization and to make the calls. This step involves selecting a panel of normals, which is a list of target counts files to be used for reference-based median normalization.

You can run the analysis in other workflow combinations. The CNV events are called for the reference samples used. Ideally the panel of normals samples follow library prep and sequencing workflows that are identical to the workflows of the case sample under analysis. For calling on sex chromosomes, the panel of normals should include a balance of both male and female samples. DRAGEN automatically handles calling on sex chromosomes based on the predicted sex of each sample in the panel.

For optimal bias correction, a minimum of 50 samples is recommended as a panel. DRAGEN can run with a single-sample panel, but single-sample panels can result in artifactual calls in the test sample where the panel sample has copy number changes.

To generate a Panel of Normals (PON), create a plain text file in which each line in the file contains a path pointing to a `.target.counts.gz` or `.target.counts.gc-corrected.gz` file generated from the target counts stage. Relative paths are supported if the paths are relative to the current working directory. Absolute paths are recommended in case the workflow is used later or shared with other users.

The following is an example PON file, which uses a subset of the GC corrected files from the target counts stage.

```
/data/output_trio1/sample1.target.counts.gc-corrected.gz
/data/output_trio1/sample2.target.counts.gc-corrected.gz
/data/output_trio2/sample4.target.counts.gc-corrected.gz
/data/output_trio2/sample5.target.counts.gc-corrected.gz
/data/output_trio3/sample7.target.counts.gc-corrected.gz
/data/output_trio3/sample8.target.counts.gc-corrected.gz
....
```

Alternatively, you can specify the files to be used in the panel of normals with the `--cnv-normals-file` option. This option uses a single file name and can be specified multiple times.

After you have created a PON file, you can run the caller by specifying your case sample with the `--cnv-input` option and the PON file with the `--cnv-normals-list` option. If you used GC bias corrected counts in the previous stage, then you do not need to run GC bias correction again. To disable GC bias correction, set `--cnv-enable-gcbias-correction` to false.

For example, the following command normalizes the case sample against the panel of normals.

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
--enable-cnv true \
--cnv-input <CASE_COUNTS> \
--cnv-normals-list <NORMALS> \
--cnv-enable-gcbias-correction false
```

Normalization Options

These options control the preconditioning of the panel of normals and the normalization of the case sample.

Option	Description
<code>--cnv-enable-self-normalization</code>	Enable/disable self-normalization mode, which does not require a panel of normals.
<code>--cnv-extreme-percentile</code>	Specifies the extreme median percentile value at which to filter out samples. The default is 2.5.
<code>--cnv-input</code>	Specifies a target counts file for the case sample under analysis when using a panel of normals.
<code>--cnv-normals-file</code>	Specifies a <code>target.counts.gz</code> file to be used in the panel of normals. You can use this option multiple times, one time for each file.
<code>--cnv-normals-list</code>	Specifies a text file that contains paths to the list of reference target counts files to be used as a panel of normals. Absolute paths are recommended in case the workflow is used later or shared with other users. Relative paths are supported if the paths are relative to the current working directory.
<code>--cnv-max-percent-zero-samples</code>	Specifies the number of zero coverage samples allowed for a target. If the target exceeds the specified threshold, then the target is filtered out. The default value is 5%. The option is sensitive to the number of normal samples being used. Make sure you adjust the threshold accordingly. If your panel of normals size is small and the threshold not adjusted, the option could filter out targets that were not intended to be.
<code>--cnv-max-percent-zero-targets</code>	Specifies the number of zero coverage targets allowed for a sample. If sample exceeds the specified threshold, then the sample is filtered out. The default value is 2.5%. The option is sensitive to the total number of target intervals. Make sure you adjust the threshold accordingly. If the capture kit has a small number of probes and the threshold not adjusted, the option could filter out targets that were not intended to be.
<code>--cnv-target-factor-threshold</code>	Specifies the bottom percentile of panel of normals medians to filter out useable targets. The default is 1% for whole genome processing and 5% for targeted sequencing processing.
<code>--cnv-truncate-threshold</code>	Specifies a percentage threshold for truncating extreme outliers. The default is 0.1%.

Segmentation

After a case sample has been normalized, the sample goes through a segmentation stage. DRAGEN implements multiple segmentation algorithms, including the following algorithms:

- Circular Binary Segmentation (CBS)
- Shifting Level Models (SLM)

The SLM algorithm has three variants, SLM, Heterogeneous SLM (HSLM), and Adaptive SLM (ASLM). HSLM is for use in exome analysis and handles target capture kits that are not equally spaced. ASLM includes additional sample-specific estimation of technical variability of depth of coverage, as opposed to changes in copy number. The estimations are based on the median variance within fixed windows or a preliminary set of segments based on b-allele ratios. The ASLM algorithm mitigates over segmentation due to noisy or wavy samples.

By default, SLM is the segmentation algorithm for germline whole genome processing, ASLM is the algorithm for somatic whole genome processing, and HSLM is the algorithm for whole exome processing.

For the targeted sequencing workflows, you can also run with a `--cnv-segmentation-bed`. The option predefines the segments to estimate copy numbers for and skips the segmentation step of the workflow. See [Targeted Segmentation \(Segment BED\) on page 194](#)

Option	Description
<code>--cnv-segmentation-mode</code>	Specifies the segmentation algorithm to perform. The following values are available: <ul style="list-style-type: none"> • <code>bed</code> • <code>cbs</code> • <code>slm</code>—The default for germline WGS analysis. • <code>aslm</code>—The default for somatic WGS analysis. • <code>hslm</code>—The default for targeted/WES analysis.
<code>--cnv-merge-distance</code>	Specifies the maximum number of base pairs between two segments that would allow them to be merged. The default value is 0 for germline WGS, which means the segments must be directly adjacent. For WES analysis, this parameter is disabled by default due to the spacing of targeted intervals.
<code>--cnv-merge-threshold</code>	Specifies the maximum segment mean difference at which two adjacent segments should be merged. The segment mean is represented as a linear copy ratio value. The default is 0.2 for germline WGS and 0.4 for WES. To disable merging, set the value to 0.

Circular Binary Segmentation

Circular Binary Segmentation is implemented directly in DRAGEN and is based on *A faster circular binary segmentation for the analysis of array CGH data*¹ with enhancements to improve sensitivity for NGS data.

The following options control Circular Binary Segmentation.

Option	Description
--cnv-cbs-alpha	Specifies the significance level for the test to accept change points. The default is 0.01.
--cnv-cbs-eta	Specifies the Type I error rate of the sequential boundary for early stopping when using the permutation method. The default is 0.05.
--cnv-cbs-kmax	Specifies maximum width of smaller segment for permutation. The default is 25.
--cnv-cbs-min-width	Specifies the minimum number of markers for a changed segment. The default is 2.
--cnv-cbs-nmin	Specifies the minimum length of data for maximum statistic approximation. The default is 200.
--cnv-cbs-nperm	Specifies the number of permutations used for p-value computation. The default is 10000.
--cnv-cbs-trim	Specifies the proportion of data to be trimmed for variance calculations. The default is 0.025.

¹ VenkatramanES, Olshen AB. A faster circular binary segmentation algorithm for the analysis of array CGH data. Bioinformatics. 2007;23(6):657-663. doi:10.1093/bioinformatics/btl646

Shifting Level Models Segmentation

The Shifting Level Models (SLM) segmentation mode follows from the R implementation of *SLMSuite: a suite of algorithms for segmenting genomic profiles*² .

Option	Description
--cnv-slm-eta	Baseline probability that the mean process changes its value. The default is 4e-5.
--cnv-slm-fw	Minimum number of data points for a CNV to be emitted. The default is 0, which means segments with one design probe could in effect be emitted.
--cnv-slm-omega	Scaling parameter that modulates relative weight between experimental or biological variance. The default is 0.3.
--cnv-slm-stepeta	Distance normalization parameter. The default is 10000. This option is only valid for HSLM.

Regardless of the segmentation method, initial segments are split across large gaps where depth data are unavailable, such as across centromeres.

² OrlandiniV, Provenzano A, Giglio S, Magi A. SLMSuite: a suite of algorithms for segmenting genomic profiles. BMC Bioinformatics. 2017;18(1). doi:10.1186/s12859-017-1734-5

Targeted Segmentation (Segment BED)

In applications for targeted panels, you can limit the segmentation and calling performed on intervals by specifying a --cnv-segmentation-bed. For example, the specified intervals might correspond to gene boundaries matched to the targeted assay. This segmentation mode is only supported with the panel of normals and requires an accompanying --cnv-target-bed. Also specify the --cnv-segmentation-bed during the panel of normals generation step, so that all interval boundaries during analysis are matched. For more information on panel of normals generation, see [Panel of Normals on page 188](#).

The recommended format for the BED file includes four columns and a header. The four columns are contig, start, stop, and name. The name column represents the name of the gene and must be unique within the BED file. The name is used in the output VCF and annotated as a segment identifier in the INFO/SEGID field. The following example file is in the recommended format:

```
contig start stop name
chr1 40356094 40372764 MYCL1
chr1 115245083 115261621 NRAS
chr1 204485504 204526342 MDM4
chr2 16075981 16090656 MYCN
chr2 29416087 30143527 ALK
chr3 12626010 12704516 RAF1
chr3 138374228 138478187 PIK3CB
chr3 178866307 178952154 PIK3CA
chr3 195776751 195806640 TFRC
```

If using a three-column BED file, then do not include a header or the name field values. Three-column BED files should only include the contig, start, and stop values. In this case, the segment identifier is autogenerated from the coordinate fields.

Quality Scoring

Quality scores are computed using a probabilistic model that uses a mixture of heavy tailed probability distributions (one per integer copy number) with a weighting for event length. Noise variance is estimated. The output VCF contains a Phred-scaled metric that measures confidence in called amplification (CN > 2 for diploid locus), deletion (CN < 2 for diploid locus), or copy neutral (CN=2 for diploid locus) events.

The scoring algorithm also calculates exact copy-number quality scores that are inputs to the DeNovo CNV detection pipeline.

Exclude BED Filtering

You can input an exclude BED to the CNV caller to filter out regions from analysis. Inputting an exclude bed is useful if there are certain regions in the genome that are known to be problematic due to library prep, sequencing, or mapping issues. You can also exclude larger intervals that specify common CNVs to aid in downstream analysis. You can create an exclude BED file using `cnv-exclude-bed`. DRAGEN does not provide an exclude BED. The intervals to exclude should be formatted in standard three-column BED format.

The intervals in the exclude BED are compared with the original target counts intervals. If the overlap is greater than `cnv-exclude-min-overlap`, the target counts interval are excluded from analysis. The `*.target.counts.gz` file still includes the interval, so you can inspect the original read counts. The normalization stage removes intervals. The `*.tn.tsv.gz` files excludes the removed intervals.

An excluded interval does not guarantee that a CNV call does not span the interval. If there is sufficient data flanking the region, the segmentation stage along with any merging might still generate a call spanning the excluded interval. However, the call would not take read counts from excluded intervals into account. You can view explanations for excluded intervals in the `*.excluded_intervals.bed.gz` file.

Output Files

The DRAGEN Host Software generates many intermediate files. `*.seg.called.merged` is the final call file that contains the amplification and deletion events.

In addition to the segment file, DRAGEN emits the calls in the standard VCF format. By default, the VCF file includes only copy number gain and loss events. For copy neutral segments, refer to the `*.seg.called.merged` file. To include copy neutral (REF) calls in the output VCF, set `--cnv-enable-ref-calls` to true.

For more information on the `*.seg.called.merged` file, and how to use the output files to aid in debug and analysis, see [Signal Flow Analysis on page 178](#).

CNV VCF File

The CNV VCF file follows the standard VCF format. Due to the nature of how CNV events are represented versus how structural variants are represented, not all fields are applicable. In general, if more information is available about an event, then the information is annotated. Some fields in the DRAGEN CNV VCF are unique to CNVs.

The following is an example of the header lines that are specific to CNV.

```
##fileformat=VCFv4.2
##CoverageUniformity=0.402517
##contig=<ID=1,length=249250621>
```

```

##contig=<ID=2,length=243199373>
##contig=<ID=3,length=198022430>
##contig=<ID=4,length=191154276>
##contig=<ID=5,length=180915260>
...
##reference=file:///reference_genomes/Hsapiens/hs37d5/DRAGEN
##ALT=<ID=CNV,Description="Copy number variant region">
##ALT=<ID=DEL,Description="Deletion relative to the reference">
##ALT=<ID=DUP,Description="Region of elevated copy number relative to the
reference"> ##INFO=<ID=REFLEN,Number=1,Type=Integer,Description="Number of REF
positions included in this record">
##INFO=<ID=SVLEN,Number=.,Type=Integer,Description="Difference in length
between REF and ALT alleles">
##INFO=<ID=SVTYPE,Number=1,Type=String,Description="Type of structural
variant">
##INFO=<ID=END,Number=1,Type=Integer,Description="End position of the variant
described in this record">
##INFO=<ID=CIPOS,Number=2,Type=Integer,Description="Confidence interval around
POS">
##INFO=<ID=CIEND,Number=2,Type=Integer,Description="Confidence interval around
END">
##FILTER=<ID=cnvQual,Description="CNV with quality below 10">
##FILTER=<ID=cnvCopyRatio,Description="CNV with copy ratio within +/- 0.2 of
1.0"> ##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=SM,Number=1,Type=Float,Description="Linear copy ratio of the
segment mean">
##FORMAT=<ID=CN,Number=1,Type=Integer,Description="Estimated copy number">
##FORMAT=<ID=BC,Number=1,Type=Integer,Description="Number of bins in the
region">
##FORMAT=<ID=PE,Number=2,Type=Integer,Description="Number of improperly paired
end reads at start and stop breakpoints">

```

Germline WGS CNV shall provide additional header lines including the following:

```

##FILTER=<ID=dinucQual,Description="CNV outside confident dinucleotide ranges
(gc_lower, gc_upper, ct_lower, ct_upper, ac_lower, ac_upper)=(0.25, 0.65, 0.3,
0.7, 0.3, 0.7)">
##FILTER=<ID=highCN,Description="CNV with a high CN > 6">
##FORMAT=<ID=GC,Number=1,Type=Float,Description="GC percentage">
##FORMAT=<ID=CT,Number=1,Type=Float,Description="CT percentage">
##FORMAT=<ID=AC,Number=1,Type=Float,Description="AC percentage">

```

The POS column is the start position of the variant. According to the VCF specification, if any of the ALT alleles is a symbolic allele, such as , then the padding base is required and POS denotes the coordinate of the base preceding the polymorphism. All coordinates in the VCF are 1-based.

The ID column is used to represent the event. The ID field encodes the event type and coordinates of the event.

The REF column contains an N for all CNV events.

The ALT column specifies the type of CNV event. Because the VCF contains only CNV events, only the DEL or DUP entry is used.

The QUAL column contains an estimated quality score for the CNV event, which is used in hard filtering.

The FILTER column contains PASS if the CNV event passes all filters, otherwise the column contains the name of the failed filter.

- Germline WGS CNV has precision filters including:
 - highCN which indicates a CNV call with implausible copy number (>6)
 - dinucQual which indicates a CNV call where some of its dinucleotide percentages are outside typical ranges, and thus the call is likely to be a false positive.

The INFO column contains information representing the event. The REFLEN entry indicates the length of the event. The SVTYPE entry is always CNV. The END entry indicates the end position of the event. If using a segment BED file, then the segment identifier is carried over from the input to SEGID field.

The FORMAT fields are described in the header.

- GT—Genotype
- SM—Linear copy ratio of the segment mean
- CN—Estimated copy number
- BC—Number of bins in the region
- PE—Number of improperly paired end reads at start and stop breakpoints

Because germline copy number calling determines overall copy number rather than the copy number on each haplotype, the genotype type field contains missing values for diploid regions when CN is greater than or equal to 2. The following are examples of the GT field for various VCF entries:

Diploid or Haploid?	ALT	FORMAT:CN	FORMAT:GT
Diploid	.	2	./.
Diploid	<DUP>	> 2	./1
Diploid		1	0/1
Diploid		0	1/1
Haploid	.	1	0
Haploid	<DUP>	> 1	1
Haploid		0	1

Coverage Uniformity

The DRAGEN CNV pipeline provides a measure of the quality of the data for a sample. If using the WGS self-normalization method, the additional CoverageUniformity metric is present in the VCF header. The metric is only available for germline samples. The CNV pipeline assumes that post-normalization target counts are independently and identically distributed (IID). Coverage in most high-quality WGS samples is uniform enough for the CNV caller to produce accurate calls, but some samples violate the IID assumption. Issues during library preparation or sample contamination can lead to several extreme outliers and/or waviness of target counts, which can result in a large number of false positive CNV calls. The CoverageUniformity metric quantifies the degree of local coverage correlation in the sample to help identify poor-quality samples.

A larger value for this metric means the coverage in a sample is less uniform, which indicates that the sample has more nonrandom noise, and could be considered poor quality. The CoverageUniformity metric depends on factors other than sample quality, such as the *cnv-interval-width* setting and sample mean coverage. DRAGEN recommends using this score to compare the quality of samples from similar mean coverage and the same command line options. Because of this, DRAGEN CNV only provides the metric and does not take any action based on it.

CNV Metrics File

DRAGEN CNV outputs metrics in CSV format. The output follows the general convention for QC metrics reporting in DRAGEN. The CNV metrics are output to a file with a `*.cnv_metrics.csv` file extension. The following list summarizes the metrics that are output from a CNV run.

Sex Genotyper Metrics

- Estimated sex karyotype for the sample, along with a confidence metric ranging from 0.0 to 1.0. If the sample sex is specified, this metric is 0.0.
- If using a panel of normals, all panel samples are also reported.

CNV Summary Metrics

- Bases in reference genome in use.
- Average alignment coverage over genome.
- Number of alignment records processed.
- Number of filtered records (total).
- Number of filtered records (due to duplicates).
- Number of filtered records (due to MAPQ).
- Number of filtered records (due to being unmapped).
- Number of target intervals.

- Number of normal samples.
- Number of segments.
- Number of amplifications.
- Number of deletions.
- Number of PASS amplifications.
- Number of PASS deletions.

Visualization and BigWig Files

To perform analysis on a known truth set, you can use the intermediate output files from the pipeline stages. These files can be parsed to aid in fine-tuning options.

All files have a structure similar to a BED file with an optional header line.

Option	Description
<code>*.target.counts.gz</code>	Contains the number of read counts per target interval. This is the raw signal as extracted from the alignments of the BAM or CRAM file. The format is identical for both the case sample and any panel of normal samples. There is also a <code>target.counts.diploid</code> file, which is normalized to the normal ploidy level of 2 instead of raw counts.
<code>*.tn.tsv.gz</code>	The tangent normalized signal of the case sample, per target interval. This file contains the log-normalized copy ratio signal. A strong signal deviation from 0.0 indicates a potential for a CNV event.
<code>*.seg.called.merged</code>	Contains the segments produced from the segmentation algorithm.
<code>*.cnv.vcf.gz</code>	Output CNV VCF file that indicates events.

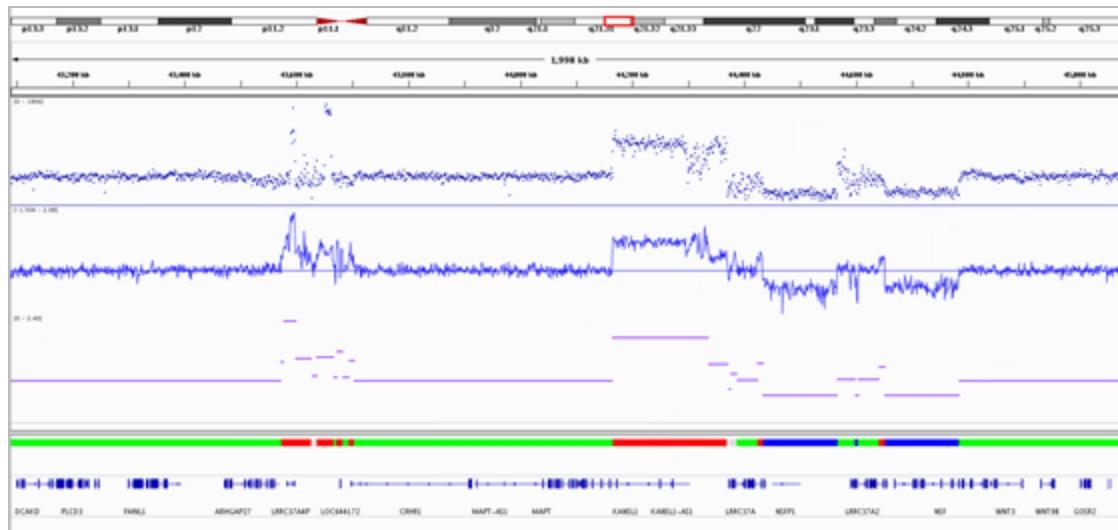
To generate additional equivalent bigWig and gff files, set the `--enable-cnv-tracks` option to true. These files can be loaded into IGV along with other tracks that are available, such as RefSeq genes. Using these tracks alongside publicly available tracks allows for easier interpretation of calls. DRAGEN autogenerates IGV session XML file if tracks are generated by DRAGEN CNV. The `*.cnv.igv_session.xml` can be loaded directly into IGV for analysis.

The following IGV tracks are automatically populated in the output IGV session file:

Option	Description
<code>.target.counts.bw</code>	BigWig representation of the target counts bins. Setting the track view in IGV to barchart or points is recommended.

Option	Description
*. <i>improper_pairs.bw</i>	BigWig representation of the improper pairs counts. Setting the track view in IGV to barchart is recommended.
*. <i>tn.bw</i>	BigWig representation of the tangent normalized signal. Setting the track view in IGV to points is recommended.
*. <i>seg.bw</i>	BigWig representation of the segments. Setting the track view in IGV to points is recommended.
*. <i>cnv.gff3</i>	GFF3 representation of the CNV events. DEL events show as blue and DUP events show as red. Filtered events are a light gray. Selecting an event brings up a window for viewing annotation details.

Figure 8 IGV Example



IGV Session XML

The IGV session XML file is prepopulated with track files generated by DRAGEN. The session file loads the reference genome that best matches the standard reference genomes in an IGV installation, by comparing the name of the `--ref-dir` specified on the command-line. Standard UCSC human reference genomes are autodetected, but any variations from the standard reference genomes might not be autodetected. To edit the genome detection, alter the `genome` attribute in the `Session` element to the reference genome you would like for analysis before loading into IGV. The reference identifier used by IGV might differ from the actual name of the genome.

i IGV may come prepacked with different flavors of GRCh37 depending on the IGV version installed. As a result, the reference naming conventions may have changed and you will have to edit the genome field in the XML file directly.

For example, IGV packages a b37 reference genome, but may also include a 1kg_v37 appearing in the IGV user interface as 1kg, b37, or a 1kg_b37+decoy appearing in the IGV user interface as 1kg, b37+decoy.

You can determine the correct encoding of a reference genome by going to `File > Save Session...` and inspecting the generated `igv_session.xml` file.

The following is an example edited session file.

```
<?xml version="1.0" encoding="utf-8"?>
<Session genome="b37" hasGeneTrack="false" hasSequenceTrack="true" version="8">
<Resources>
<Resource path="example.cnv.gff3"/>
<Resource path="example.cnv.excluded_intervals.bed.gz"/>
<Resource path="example.target.counts.bw"/>
<Resource path="example.improper.pairs.bw"/>
<Resource path="example.tn.bw"/>
<Resource path="example.seg.bw"/>
</Resources>
<Panel height="500" width="1200" name="DataPanel">
...
</Panel>
</Session>
```

Exclude Interval Files

To improve accuracy, the DRAGEN CNV Pipeline excludes genomic intervals if one or more the target intervals failed at least one quality requirement. The excluded intervals are reported to `*.excluded_intervals.bed.gz` file. The file identifies the regions of the genome that are not callable for CNV analysis and describes the reason intervals were excluded in the fourth column. The following are the possible reasons for exclusion.

Exclusion Reason	Description	Command Line Option
NON_KMER_UNIQUE	Nonunique Kmer bases are larger than 50% of interval.	Not applicable. This reason only applies to self-normalization mode.
EXCLUDE_BED	Interval overlaps with exclude BED larger than threshold.	--cnv-exclude-min-overlap
PON_MAX_PERCENT_ZERO_SAMPLES	Number of PON samples with 0 coverage is larger than threshold.	--cnv-max-percent-zero-samples
PON_TARGET_FACTOR_THRESHOLD	Median coverage of interval is lower than threshold of overall median coverage.	--cnv-target-factor-threshold
PON_MISSING_INTERVAL	Target interval not found in PON.	Not applicable.

Output and Filtering Options

The output and filtering options control the CNV output files.

Option	Description
--cnv-exclude-bed	Specifies a BED file that indicates the intervals to exclude from the CNV analysis. If a target interval overlaps regions specified from exclude BED file more than <i>cnv-exclude-min-overlap</i> , the target interval is suppressed.
--cnv-exclude-min-overlap	Specifies a fraction for filtering threshold of overlap amount between a target interval and the excluded region (0.5).
--cnv-enable-ref-calls	Emit copy neutral (REF) calls in the output VCF file. The default is true for single WGS CNV analysis.
--cnv-enable-tracks	Generate track files that can be imported into IGV for viewing. When this option is enabled, a * .gff file for the output variant calls is generated, as well as * .bw files for the tangent normalized signal. The default is true.
--cnv-filter-bin-support-ratio	Filters out a candidate event if the span of supporting bins is less than the specified ratio with respect to the overall event length, and only applies to records with length greater than 80kbp. The default ratio is 0.2 (20% support). As an example, if an event is called and has a length of 100,000 bp, but the target interval bins that support the call only spans a total of 15,000 bp ($15,000/100,000 = 0.15$), then the interval is filtered out.
--cnv-filter-copy-ratio	Specifies the minimum copy ratio threshold value centered about 1.0 at which a reported event is marked as PASS in the output VCF file. The default value is 0.2 that leads to calls less than CR=0.8 or greater than CR=1.2.
--cnv-filter-length	Specifies the minimum event length in bases at which a reported event is marked as PASS in the output VCF file. The default is 10000.
--cnv-filter-qual	Specifies the QUAL value at which a reported event is marked as PASS in the output VCF file. You should adjust the parameter value according to your own application data.
--cnv-min-qual	Specifies the minimum reported QUAL. The default is 3.
--cnv-max-qual	Specifies the maximum reported QUAL. The default is 200.
--cnv-qual-length-scale	Specifies the bias weighting factor to adjust QUAL estimates for segments with longer lengths. This is an advanced option and should not need to be modified. The default is 0.9303 (2-0.1).
--cnv-qual-noise-scale	Specifies the bias weighting factor to adjust QUAL estimates based on sample variance. This is an advanced option and should not need to be modified. The default is 1.0.

Concurrent CNV and Small Variant Calling

DRAGEN can perform mapping and aligning of FASTQ samples, and then directly stream the data to downstream callers. If the input is a FASTQ sample, a single sample can run through both the CNV and the small VC. This triggers self-normalization by default.

Run the FASTQ sample through a regular DRAGEN map/align workflow, and then provide additional arguments to enable the CNV, VC, or both. The options that apply to CNV in the standalone workflows are also applicable here.

The following examples show different commands.

Map/Align FASTQ With CNV

```
dragen \
-r <HASHTABLE> \
-1 <FASTQ1> \
-2 <FASTQ2> \
--RGSM <SAMPLE> \
--RGID <RGID> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align true \
--enable-cnv true \
--cnv-enable-self-normalization true
```

Map/Align FASTQ With VC

```
dragen \
-r <HASHTABLE> \
-1 <FASTQ1> \
-2 <FASTQ2> \
--RGSM <SAMPLE> \
--RGID <RGID> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align true \
--enable-variant-caller true
```

Map/Align FASTQ With CNV and VC

```
dragen \
-r <HASHTABLE> \
-1 <FASTQ1> \
-2 <FASTQ2> \
--RGSM <SAMPLE> \
--RGID <RGID> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align true \
--enable-cnv true \
--cnv-enable-self-normalization true \
--enable-variant-caller true
```

BAM Input to CNV and VC

```
dragen \
-r <HASHTABLE> \
--bam-input <BAM> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
--enable-cnv true \
--cnv-enable-self-normalization true \
--enable-variant-caller true
```

Sample Correlation and Sex Genotyper

When running the target counts stage or the normalization stage, the DRAGEN CNV pipeline also provides the following information about the samples in the run.

- A correlation metric of the read count profile between the case sample and any panel of normals samples. A correlation metric greater than 0.90 is recommended for confident analysis, but there is no hard restriction enforced by the software.
- The predicted sex of each sample in the run. The sex is predicted based on the read count information in the sex chromosomes and the autosomal chromosomes. The median value for the counts is printed to the screen for the autosomal chromosomes, the X chromosome, and the Y chromosome.

The results are printed to the screen when running the pipeline. For example:

```
=====
Correlation Table
=====
```

```
Correlation of case sample PlatinumGenomes_50X_NA12877 against
PlatinumGenomes_50X_NA12878: 0.984092
```

Sex Genotyper

Predicted sex of samples

```
PlatinumGenomes_50X_NA12877: MALE XY 0.99737
```

```
PlatinumGenomes_50X_NA12878: FEMALE XX 0.968929
```

The predicted sexes for samples in use are also printed to the `*.cnv_metrics.csv` output file.

For a panel of normals, the predicted sexes are used to determine which panel samples are leveraged for normalization on sex chromosomes. If the estimated sex of the sample is UNDETERMINED, the sex of the sample is set to FEMALE.

You can override the predicted sex of the sample with the `--sample-sex` option.

Allele Specific CNV for Somatic WES CNV

To detect somatic copy number aberrations and regions with loss of heterozygosity, run the DRAGEN CNV Caller on a tumor sample with a VCF that contains germline SNVs. The output file is a VCF file. Components of the germline CNV caller are reused in the somatic algorithm with the addition of a somatic modeling component, which estimates tumor purity and ploidy.

The germline SNVs are used to compute b-allele ratios in the tumor, which allows for allele-specific copy number calling on the tumor sample. Where possible, use of the small-variant VCF from a matched normal sample.

Panel of normals are used for the reference baseline to provide insight into copy number variants. The ASCN somatic WES CNV model is similar to the somatic WGS CNV model, but use a panel of normals to remove coverage bias in each target regions. We also apply different internal parameters tuned for WES.

Pipeline can accept various input types for matched normal sample for b-allele loci. If normal sample is already processed using the germline small variant caller, user can provide VCF file. If normal sample is not processed, then user can provide raw reads or aligned reads together with enabling small variant caller, then DRAGEN CNV will directly accept VCF from small variant caller, and use it for b-allele loci.

ASCN Somatic WES CNV Calling Options

You can use following somatic WES CNV calling command line options:

Input	Option	Argument	Description
Tumor input	--tumor- fastq1 --tumor- fastq2 --tumor- bam-input --tumor- cram-input	file	Specify a tumor input file.
Normal input	--fastq1 --fastq2	file	Specify a normal input file (if normal VCF is not ready).
Option 1	--bam- input --cram- input		
Normal input	--cnv-use- somatic- Option 1 vc-baf	true/false	If running in tumor-normal mode with the SNV caller enabled, use this option to specify the germline heterozygous sites. For more information on specifying b-allele loci, see Specify B-Allele Loci on page 221 .
Normal input	--cnv- normal-b- Option 2 allele-vcf	vcf file	Specify a matched normal SNV VCF. For more information on specifying b-allele loci, see Specify B-Allele Loci on page 221 .
PON option 1	--cnv- normals- file	normal count file	Specify individual normal counts file (target.counts.gz or target.counts.gc-corrected.gz) for PON. You can use this option multiple times, one time for each file.
PON option 2	--cnv- normals- list	text file indicating normal count files per line	Specify text file that contains paths to the list of reference target counts files to be used as a panel of normals (new line separated).

Input	Option	Argument	Description
PON option 3	--cnv-combined-counts	file	Specify combined PON file (.combined.counts.txt.gz).
PON option 4	Not applicable		If no PON sample is specified, then DRAGEN uses matched normal sample as single sample PON. Available for Normal input Option 1.
Target region	--cnv-target-bed	bed file	Specify target region bed file
Sample sex	--sample-sex	male/female/auto/none	If known, specify the sex of the sample. If the sample sex is not specified, the caller attempts to estimate the sample sex from tumor alignments.

Input requirements:

- 1 tumor input
- 1 normal input (either option 1 or 2)
- Panel of normals (either option 1, 2, 3 or 4)
- Target region

When the normal sample input is not in VCF format (eg, FASTQ/BAM/CRAM), then the normal sample shall be capable of being used as PON. However, if the normal sample is already included in the PON, then it will not be added.

Example command lines

The following is an example command line for running ASCN tumor-normal somatic WES CNV calling with matched normal SNV VCF.

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
--enable-cnv true \
--tumor-bam-input <TUMOR_BAM> \
--cnv-normal-b-allele-vcf <SNV_NORMAL_VCF> \
```

```
--cnv-normals-list <PANEL_OF_NORMALS> \
--cnv-target-bed <TARGET_BED> \
--sample-sex <SEX>
```

The following example command line runs ASCN tumor-normal somatic WES CNV calling concurrently with the Somatic SNV Caller, which allows you to use the matched normal germline heterozygous sites directly from the SNV Caller with the command `cnv-use-somatic-vc-baf true`.

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
--tumor-bam-input <TUMOR_BAM> \
--bam-input <NORMAL_BAM> \
--enable-cnv true \
--enable-variant-caller true \
--cnv-use-somatic-vc-baf true \
--cnv-normals-list <PANEL_OF_NORMALS> \
--cnv-target-bed <TARGET_BED> \
--vc-target-bed <TARGET_BED> \
--sample-sex <SEX>
```

Method and Outputs

ASCN somatic WES CNV pipeline uses the same methods and workflow as the DRAGEN Somatic WGS CNV pipeline. Refer to [Somatic WGS CNV Calling on page 218](#) for more information.

CNV with SV Support

The DRAGEN CNV caller leverages depth as its primary signal for calling copy number variants. Depth alone poses challenges for calling events that are less than 10kbp. The sensitivity of CNVs at lengths less than 10kbp can be improved by leveraging junction signals from the DRAGEN structural variant caller.

When both the DRAGEN CNV and SV caller are executed in a single invocation, then an additional integration step is done at the end of a DRAGEN run to improve the CNV calls. This feature is enabled automatically when DRAGEN detects a germline WGS analysis.

The SV/CNV Integration module takes in DEL and DUP calls from the output data structures of the germline CNV and SV callers, identifies putative matches, updates annotations, filters, scores, and outputs the refined records in a new output VCF. By leveraging junction signals from the SV caller and depth signals from the CNV caller, this approach allows for sensitive CNV detection down to 1kbp while

also improving recall and precision across length scales. This is achieved by rescuing previously low quality calls if evidence is found from both callers, and also by adjusting CNV breakends to the more accurate SV breakends. The matching algorithm takes into account the proximity of the events as well as the transition states at the breakends, among other things.

Example command lines

The following is an example command line for running a germline WGS analysis for both CNV and SV.

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--ref-dir <HASHTABLE> \
--bam-input <BAM> \
--enable-map-align false \
--enable-cnv true \
--cnv-enable-self-normalization true \
--enable-sv true \
```

Other optional CNV or SV parameters can also be added.

Combined CNV/SV VCF Output

The original CNV and SV VCF output files will still be available for users in the DRAGEN output directory. Additionally, there will be an enhanced CNV VCF available with the *.cnv_sv.vcf.gz extension. The VCF header lines in the *.cnv_sv.vcf.gz will mostly resemble a concatenation of the individual header lines from the CNV and SV VCFs, with a few lines deduplicated and some new ones added. For details on the legacy header lines, refer to the individual CNV and SV user guide sections.

Newly added header lines are described in the following table.

Header Field	Value	Type	Description
END_LEFT_BND_OF	1	String	ID of CNV whose left end is matched to the end of SV
END_RIGHT_BND_OF	1	String	ID of CNV whose right end is matched to the end of SV
LEFT_BND	1	String	ID of SV that matches the left end of CNV record
LEFT_BND_OF	1	String	ID of CNV whose left end is matched to SV
MatchSv	1	Integer	ID of original SV that was merged with CNV record

Header Field	Value	Type	Description
OrigCnvEnd	1	Integer	Coordinate of original CNV end
OrigCnvPos	1	Integer	Coordinate of original CNV pos
RIGHT_BND	1	String	ID of SV that matches the right end of CNV record
RIGHT_BND_OF	1	String	ID of CNV whose right end is matched to SV
SVCLAIM	A	String	Claim made by the structural variant call. Valid values are D, J, DJ for abundance, adjacency and both respectively

Records that can be matched or rescued will have annotations indicating the breakpoint linkage between a CNV and SV record. If a complete match is found, then the MatchSv annotation will be present in the record, indicating the SV record's ID field for this CNV record. Furthermore, the use of the SVCLAIM field will indicate if the record has evidence arising from depth signal D, or junction signals J, or both DJ.

Because of the mixing of standalone SV records and CNV records, the FORMAT field maybe have different annotations. This is allowed by the VCF specification as the FORMAT field is defined per record. For details on the CNV or SV specific annotations, please refer to the individual CNV and SV user guide sections.

Most FILTERs will still apply to individual records, with joint records being set to PASS. For example, the cnvLength FILTER will still be applied to standalone CNV records (those with SVCLAIM=D).

Example records are shown below:

```
# Merged record, note presence of SVCLAIM=DJ and MatchSv
1 869444 DRAGEN:LOSS:1:869444-870284 N <DEL> 150 PASS SVLEN=-
840;SVTYPE=CNV;END=870284;REFLEN=840;OrigCnvPos=869000;OrigCnvEnd=871000;SVCLAI
M=DJ;MatchSv=DRAGEN:DEL:41710:0:0:0:2:0 GT:SM:CN:BC:GC:CT:AC:PE
1/1:0.649442:1:2:0.6785:0.408:0.3705:10,3

# CNV record that did not match, note presence of SVCLAIM=D
1 13472000 DRAGEN:LOSS:1:13472001-13663000 N <DEL> 69 PASS SVLEN=-
191000;SVTYPE=CNV;END=13663000;REFLEN=191000;SVCLAIM=D GT:SM:CN:BC:GC:CT:AC:PE
0/1:0.427273:1:141:0.467603:0.501092:0.498667:7,10

# SV record that did not match, note presence of SVCLAIM=J
1 14657708 DRAGEN:LOSS:1:14657708-14658485 N <DEL> 150 PASS
END=14658485;SVTYPE=DEL;SVLEN=-
776;CIGAR=1M776D;CIPOS=0,2;HOMLEN=2;HOMSEQ=TC;SVCLAIM=J GT:FT:GQ:PL:PR:SR
0/1:PASS:671:908,0,668:36,13:27,18
```

Multisample CNV Calling

Multisample CNV calling is possible starting from tangent normalized counts files (*.tn.tsv.gz) specified with the `--cnv-input` option (one per sample). Multisample CNV analysis benefits from using joint segmentation to increase the sensitivity of detection of copy number variable segments. For each copy number variable segment identified, the copy number genotype of each sample is emitted in a single VCF entry to facilitate annotation and interpretation.

Multisample CNV calling is supported for WGS and WES workflows.

The following is an example command line for running a trio analysis:

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-cnv true \
--cnv-input <FATHER_TN_TSV> \
--cnv-input <MOTHER_TN_TSV> \
--cnv-input <PROBAND_TN_TSV> \
--pedigree-file <PEDIGREE_FILE>
```

De Novo CNV Calling Options

Make sure all input samples have gone through the same single sample workflow and have identical intervals. If the samples are WES inputs, then you must generate the samples using the same panel of normals, and the autosomal intervals for all samples must match.

The following options are used in DeNovo CNV calling:

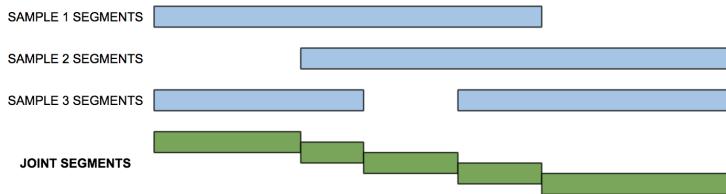
Option	Description
<code>--cnv-input</code>	For DeNovo CNV calling, this specifies the input tangent-normalized signal files (*.tn.tsv) from the single sample runs. This option can be specified multiple times, once for each input sample.
<code>--cnv-input</code>	For DeNovo CNV calling, this specifies the input tangent-normalized signal files (*.tn.tsv) from the single sample runs. This option can be specified multiple times, once for each input sample.

Option	Description
--cnv-filter-de-novo-qual	Phred-scaled threshold at which a putative event in the proband sample is marked as DeNovo. Default value is 0.125.
--pedigree-file	Pedigree file specifying the relationship between the input samples.

Joint Segmentation

First, CNV calling is performed on each sample independently. Joint segmentation then uses the copy number variable segments from each single sample analysis to derive a set of joint copy number variable segments. This set of joint segments is determined simply by taking the union of all breakpoints from the copy number variable segments of all samples. This results in the splitting of any partially overlapping segments across different samples.

Figure 9 Overlapping Segments



Following joint segmentation, copy number calling is again performed independently on each sample using the joint segments. Segments can be merged as with the single sample analysis, but each joint segment is emitted in the multisample VCF as a single entry. The quality score (QS in the VCF) from the sample's merged segment, if applicable, is used for filtering the call. Sample calls are filtered using the sample's FT field in the multisample VCF. The QUAL column of the multisample VCF is always missing (ie, "."). The FILTER column of the multisample VCF is "SampleFT" if none of the sample's FT fields are "PASS", and "PASS" if any of the sample's FT fields are "PASS".

De Novo Calling Stage

A de novo event is defined as the existence of a genotype at a particular locus in a proband's genome that did not result from standard Mendelian inheritance from the parents. The de novo calling stage identifies putative de novo events in the proband of each trio of a multisample analysis. In some cases, these putative de novo events may be real, but they can also arise from sequencing or analysis artifacts. Consequently, a de novo quality score is assigned to each putative de novo event and used to filter out low-quality de novo events. Trios are specified by specifying a .ped file with the --pedigree-file option. Multiple trios can be specified (eg, quad analysis), and all valid trios will be processed.

For each joint segment in a trio, the de novo caller determines if there is a Mendelian inheritance conflict for the called copy number genotypes. The CNV caller does not identify the copy number for each allele of a given diploid segment, which means assumptions are made about the possible allelic composition of the parent genotypes.

The assumption is that the copy number 0 allele is not present for diploid regions of a parent's genome (sex dependent) when the assigned copy number is 2 or greater. This results in simplifications, as follows:

Parent Copy Number Genotype	Possible Copy Number Alleles	Assumed Possible Copy Number Alleles
2	0/2, 1/1	1/1
3	0/3, 1/2	1/2
4	0/4, 1/3, 2/2	1/3, 2/2
N	x/(N-x) for x <= N/2	x/(N-x) for 1 <= x <= N/2

The following are examples of consistent and inconsistent copy number genotypes for diploid regions using these assumptions:

Mother Copy Number	Father Copy Number	Proband Copy Number	Mendelian Consistent?
2	2	2	Yes
2	2	1	No
3	2	4	No
3	2	2	Yes
2	0	2	No

If a joint segment has a Mendelian inheritance conflict, a Phred-scaled de novo quality score (DQ field in the VCF) is calculated using the likelihoods for each copy number state (see Quality Scoring section) of each sample in the trio, combined with a prior for the trio genotypes:

```
DQ = -10log (1-Sum over conflicting genotypes (p(CNm|data) *p(CNf|data) *p(CNp|data) *p(CNm,CNf,CNp)) /Sum over all genotypes (p(CNm|data) *p(CNf|data) *p(CNp|data) *p(CNm,CNf,CNp)))
```

Where

- CNm = Mother copy number
- CNf = Father copy number
- CNp = Proband copy number
- $p(CNm, CNf, CNp)$ = the prior for the trio genotype

The DN field in the VCF is used to indicate the de novo status for each segment. Possible values are:

- Inherited - the called trio genotype is consistent with Mendelian inheritance

- LowDQ - the called trio genotype is inconsistent with Mendelian inheritance and DQ is less than the de novo quality threshold (default 0.125)
- DeNovo - the called trio genotype is inconsistent with Mendelian inheritance and DQ is greater than or equal to the de novo quality threshold (default 0.125)

Multisample CNV VCF Output

The records in a multisample CNV VCF differ slightly from the single sample case. The major differences are as follows:

- The per-record entries are broken down into the segments among the union of all the input samples breakpoints, which means there are more entries in the overall VCF.
- The QUAL column is not used and its value is “.”. The per-sample quality is carried over into the SAMPLE columns with the QS tag.
- The FILTER column indicates PASS if any of the individual SAMPLE columns PASS. Otherwise, it indicates SampleFT.
- The per-sample annotations are carried over from their originating calls. The single sample filters are applied at the sample level and are emitted in the FT annotation.

Additionally, if a valid pedigree is used, then de novo calling is performed, which adds the following two annotations to the proband sample.

```
##FORMAT=<ID=DQ,Number=1,Type=Float,Description="De novo quality">
##FORMAT=<ID=DN,Number=1,Type=String,Description="Possible values are
`Inherited', 'DeNovo' or 'LowDQ'. Threshold for a passing de novo call is DQ >
0.125000">
```

While the VCF contains many entries, due to the joint segmentation stage, the number of de novo events can be found by extracting entries that have a DN and DQ annotation. These records are also extracted and are converted to GFF3 in the de novo calling case.

Somatic CNV Calling

Depending on your workflow, you can use the following somatic CNV callers.

- For whole-genome sequencing (WGS), use the Somatic WGS CNV Caller. For more information, see [Somatic WGS CNV Calling on page 218](#).
- For whole-exome sequencing (WES), use the Somatic WES CNV Caller. For more information, see [Somatic WES CNV Calling on page 215](#)

Somatic WES CNV Calling

For somatic whole-exome sequencing (WES) and somatic targeted panels, you can use a panel of normals as the reference baseline to provide insight into copy number variants. The reported events are based solely on the normalized copy ratio values and the deviation from the expected reference

baseline levels. This workflow can be useful for applications that require only the detection of gains and losses in targeted genes. The somatic WES CNV model is similar to the germline WES CNV model, but utilizes a different quality scoring and calling model.

Use one of the following input options.

- `--tumor-fastq1` and `--tumor-fastq2`—Specify a FASTQ file
- `--tumor-bam-input`—Specify an existing BAM file
- `--tumor-cram-input`—Specify an existing CRAM file

The Somatic WES CNV Caller requires a panel of normals. The panel of normals samples help measure intrinsic biases of the upstream processes to allow for proper normalization. To generate a panel of normals, see [Panel of Normals on page 188](#). The panel of normals sample should be well matched to the case sample under analysis.

If a matched normal sample is available, the sample can be included in the panel of normals. The workflow does not change if a matched normal is or is not available.

Example Command Lines

The following example command line runs somatic analysis on WES data.

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
--enable-cnv true \
--tumor-bam-input <TUMOR_BAM> \
--cnv-normals-list <NORMALS> \
--cnv-target-bed <BED> \
```

If you are using somatic targeted panels with a set of genes supplied with the capture kit, then you can bypass segmentation by specifying a `cnv-segmentation-bed` and using `cnv-segmentation-mode=bed`. If using this option, all events in the segmentation BED are reported in the output VCF. For more information on the segmentation BED file, see [Targeted Segmentation \(Segment BED\) on page 194](#).

The following example command line runs somatic analysis on a targeted panel.

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
```

```
--enable-cnv true \
--tumor-bam-input <TUMOR_BAM> \
--cnv-normals-list <NORMALS> \
--cnv-target-bed <BED> \
--cnv-segmentation-bed <SEGMENT_BED> \
--cnv-segmentation-mode bed \
```

Quality Scoring and Calling

The Somatic WES CNV Caller computes quality scores using a 2 sample t-test between the normalized copy ratio of the case sample and the panel of normals samples. The caller computes a p-value per segment. The p-values are then converted to Phred-scaled scores. For copy neutral events, the caller computes quality scores as $1-p$.

DUP/DEL events calls are made based on the limit of detection threshold (LoD) which is set using `cnv-filter-limit-of-detection` (default 0.2). For each segment, the caller compute a p-value for hypothetical counts by Case Counts X ($1 +/- LoD$) against PON. If p-value of Case Counts X ($1+LoD$) is highest, then segment is called as DUP. If p-value of Case Counts X ($1-LoD$) is highest, then segment is called DEL. Otherwise segment is called REF.

The output VCF contains the quality score in the `QUAL` field.

Somatic WES CNV Output

The somatic WES CNV VCF file follows standard VCF format and contains the following differences from the germline CNV VCF output. For information on the germline CNV VCF output, see [CNV VCF File on page 195](#).

The `FORMAT` fields are described in the header section and are identical to the germline CNV VCF output, except the somatic WES CNV VCF does not include the `CN` entry. The Somatic WES CNV Caller does not estimate the tumor purity fraction and cannot make an estimate of the copy number. For more information, see [Tumor Purity and Fold Change on page 218](#).

The following is an example VCF output where a segmentation BED was included in somatic WES CNV calling.

```
chr7 92243233 DRAGEN:REF:chr7:92243233-92462639 N . 3 PASS
END=92462639;REFLEN=219407;SEGID=CDK6 GT:SM:BC:PE ./.:1.0257:28:125,0
chr7 116339136 DRAGEN:GAIN:chr7:116339137-116436180 N <DUP> 200 PASS
SVLEN=97044;SVTYPE=CNV;END=116436180;REFLEN=97044;SEGID=MET GT:SM:BC:PE
./1:1.37061:36:2287,989
chr7 140434395 DRAGEN:REF:chr7:140434395-140624505 N . 3 PASS
END=140624505;REFLEN=190111;SEGID=BRAF GT:SM:BC:PE ./.:0.977961:38:73,0
```

Tumor Purity and Fold Change

Due to the sparseness of WES and targeted panels, b-allele data are insufficient for an accurate estimate of tumor purity. The Somatic WES CNV Caller only reports copy ratio, also known as fold change. Fold change is encoded in the `FORMAT/SM` field as a linear copy ratio of the segment mean.

If tumor purity is known, you can infer the ploidy of a gene or segment in the sample from the reported fold change using the following calculation.

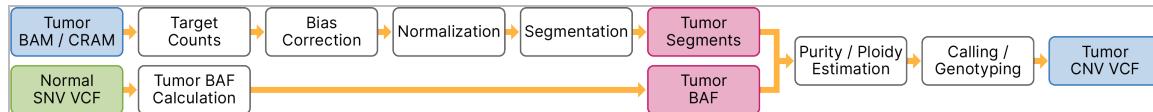
For example, if the tumor purity is 30% for MET with a fold change of 2.2x, then there are 10 copies of MET DNA in the sample.

Somatic WGS CNV Calling

To detect somatic copy number aberrations and regions with loss of heterozygosity, run the DRAGEN CNV Caller on a tumor sample with a VCF that contains germline SNVs. The output file is a VCF file. Components of the germline CNV caller are reused in the somatic algorithm with the addition of a somatic modeling component, which estimates tumor purity and ploidy.

The germline SNVs are used to compute b-allele ratios in the tumor, which allows for allele-specific copy number calling on the tumor sample. Where possible, use of the small-variant VCF from a matched normal sample is preferred (tumor-normal mode), but a catalog of population SNPs can be used when a matched normal sample is not available (tumor-only mode).

Figure 10 Somatic CNV Whole-Genome Sequencing (WGS) Caller Workflow



When a matched normal sample is available, the sample should first be processed using the germline small variant caller. In this case, only germline-heterozygous SNV sites are used for determining b-allele ratios. If no matched normal is available, population SNP b-allele ratios are computed as for matched normal heterozygous loci, but are treated as variants of unknown germline genotype; possible genotype assignments are statistically integrated to determine allele-specific copy number.

In matched normal mode, a VCF containing germline copy number changes for the individual may optionally be input. This makes sure that germline CNVs are output as separate segments in the somatic whole-genome sequencing (WGS) CNV VCF, and annotated with the germline copy number so that it is clear whether there are specifically-somatic copy number changes in the region.

Somatic WGS CNV Calling Options

You can use the following somatic WGS CNV calling command-line options:

Option	Description
<ul style="list-style-type: none"> • <code>--tumor-fastq1</code>, • <code>--tumor-fastq2</code>, • <code>--tumor-bam-input</code> • <code>--tumor-cram-input</code> 	Specify a tumor input file.
<code>--cnv-normal-b-allele-vcf</code>	Specify a matched normal SNV VCF. For more information on specifying b-allele loci, see Specify B-Allele Loci on page 221 .
<code>--cnv-population-b-allele-vcf</code>	Specify a population SNP catalog. For more information on specifying b-allele loci, see Specify B-Allele Loci on page 221 .
<code>--cnv-use-somatic-vc-baf</code>	If running in tumor-normal mode with the SNV caller enabled, use this option to specify the germline heterozygous sites. For more information on specifying b-allele loci, see Specify B-Allele Loci on page 221 .
<code>--sample-sex</code>	If known, specify the sex of the sample. If the sample sex is not specified, the caller attempts to estimate the sample sex from tumor alignments.
<code>--cnv-normal-cnv-vcf</code>	Specify germline CNVs from the matched normal sample. For more information, see Germline-Aware Mode on page 222 .
<code>--cnv-use-somatic-vc-vaf</code>	Use the variant allele frequencies (VAFs) from the somatic SNVs to help select the tumor model for the sample. For more information, see VAF-aware Mode on page 223 .
<code>--cnv-somatic-enable-het-calling</code>	Enable HET-calling mode for heterogeneous segments. For more information see, HET-calling Mode on page 223 .

The following is an example command line for running the tumor-normal somatic WGS CNV Caller with a matched normal SNV VCF.

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
--enable-cnv true \
--tumor-bam-input <TUMOR_BAM> \
--cnv-normal-b-allele-vcf <SNV_NORMAL_VCF> \
--sample-sex <SEX>
```

If a matched normal is not available, you must disable CNV calling or run in tumor-only mode. Running with a mismatched normal in tumor-normal mode yields unexpected results. The following example command line runs tumor-only somatic WGS CNV calling with a population SNV VCF.

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
--enable-cnv true \
--tumor-bam-input <TUMOR_BAM> \
--cnv-population-b-allele-vcf <SNV_POP_VCF> \
--sample-sex <SEX>
```

The following example command line runs tumor normal somatic WGS CNV calling concurrently with the Somatic SNV Caller, which allows you to use the matched normal germline heterozygous sites directly from the SNV Caller with the command `cnv-use-somatic-vc-baf true`.

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
--enable-cnv true \
--tumor-bam-input <TUMOR_BAM> \
--bam-input <NORMAL_BAM>
--enable-variant-caller true \
--cnv-use-somatic-vc-baf true \
--sample-sex <SEX>
```

You can enable additional features when a matched normal sample and the outputs from DRAGEN Germline analysis are also available. If a matched normal sample is available, enable germline-aware mode and VAF-aware mode using the following example command line. For more information on germline-aware mode and VAF-aware mode, see [Somatic WGS CNV Calling Modes on page 222](#) and [VAF Aware Mode](#).

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align false \
--enable-cnv true \
--tumor-bam-input <TUMOR_BAM> \
--bam-input <NORMAL_BAM>
--enable-variant-caller true \
```

```
--cnv-use-somatic-vc-baf true \
--cnv-normal-cnv-vcf <CNV_NORMAL_VCF> \
--sample-sex <SEX>
```

Target Counts and B-allele Counts

The target counting stage and its output are the same for the germline CNV calling case. The target intervals with the read counts are output in a *.target.counts file. If there is insufficient read depth coverage detected, processing will halt. For low depth tumor samples, the value of `--cnv-interval-width` can be increased from to capture more alignments. The b-allele counting occurs in parallel with the read counting phase, and the values are output in a *.baf.bedgraph.gz file. This file can be loaded into IGV along with other bigwig files generated by DRAGEN for visualization.

Specify B-Allele Loci

The Somatic WGS CNV Caller requires a source of heterozygous SNP sites to measure b-allele counts of the tumor sample. The following are the available modes.

Option	Description
<code>cnv-normal-b-allele-vcf</code>	Specify a matched normal SNV VCF. Use when a matched normal sample and the matched normal SNV VCF are available. To use this option, you must run the matched normal sample through the DRAGEN Germline workflow.
<code>cnv-population-b-allele-vcf</code>	Specify a population SNP VCF. Use when a matched normal sample is not available and analysis must be performed in tumor-only mode.
<code>cnv-use-somatic-vc-baf</code>	Set to <code>true</code> to bypass the DRAGEN Germline workflow. Use if tumor and matched normal input are available. Enable the Somatic SNV Caller to use this option.

To specify a matched normal sample SNV VCF, use the `--cnv-normal-b-allele-vcf` option. The VCF file should come from processing the matched normal sample through the DRAGEN germline small variant caller with filters applied. Typically, this file name has a *.hard-filtered.vcf.gz extension. All records marked as `PASS` that are determined to be heterozygous in the normal sample are used to measure the b-allele counts of the tumor sample. You can also use equivalent gVCF file (*.hard-filtered.gvcf.gz), but the processing time is significantly longer due to the number of records, most of which are not heterozygous sites.

To specify a population SNP VCF, use `--cnv-population-b-allele-vcf` option. To obtain a population SNP VCF, process an appropriate catalog of population variation, such as from dbSNP, the 1000 genome project, or other large cohort discovery efforts. A suitable example file for this parameter is `1000G_phase1.snps.high_confidence.vcf.gz` from the GATK resource bundle. Only high-frequency SNPs should be included. For example, include SNPs with minor allele population frequency $\geq 10\%$ to limit run

time impact and reduce artifacts. Specify the ALT allele frequency by adding `AF=<alt frequency>` to the `INFO` section of each record. Additional `INFO` fields might be present, but DRAGEN only parses and uses the `AF` field. Sites specified with `--cnv-population-b-allele-vcf` can be either heterozygous or homozygous in the germline genome from which the tumor genome derives.

The following is an example valid population SNP record:

```
chr1 51479 . T A 1000 PASS AF=0.3253
```

DRAGEN considers the following requirements when parsing records from the b-allele VCF:

- Only simple SNV sites.
- Records must be marked `PASS` in the `FILTER` field.
- If there are records with the same `CHROM` and `POS` in the VCF, then DRAGEN uses the first record that occurs.

If a tumor sample and matched normal input are available, use `--cnv-use-somatic-vc-baf true`. You must enable the Somatic SNV Caller. If using this option, DRAGEN determines the germline heterozygous sites from the matched normal input and measures the b-allele counts of the tumor sample. The information is passed to the Somatic WGS CNV Caller to simplify the overall somatic workflow.

To enable `--cnv-use-somatic-vc-baf`, enter the following command line options.

- `--tumor-bam-input <TUMOR_BAM>`—Specify the tumor input
- `--bam-input <NORMAL_BAM>`—Specify the matched normal input
- `--enable-variant-caller true`—Enable the somatic SNV variant caller
- `--cnv-use-somatic-vc-baf true`—Enable somatic VC BAF

Somatic WGS CNV Calling Modes

Germline-Aware Mode

To specify germline CNVs from a matched normal sample, use `--cnv-normal-cnv-vcf`. When specified, CNV records marked as `PASS` in the normal sample are used during tumor-sample segmentation to make sure that confident germline CNV boundaries are also boundaries in the somatic output. Segments with germline copy number changes that are relative to reference ploidy are excluded from somatic model selection.

During somatic copy number calling and scoring, the germline copy number is used to modify the expected depth contribution from the normal contamination fraction of the tumor sample. The process leads to more accurate assignment of somatic copy number in regions of germline CNV. DRAGEN then annotates the somatic WGS CNV VCF entries with germline copy number (NCN) and the somatic copy number difference relative to germline (SCND) for the segments that have germline CNVs.

VAF-aware Mode

If both the small variant caller and the CNV caller are enabled in a tumor-matched normal run, the somatic SNV results can affect the estimated purity and ploidy of the tumor sample. The somatic SNV variant allele frequencies (VAFs) that are captured by the allele depth values from passing somatic SNVs reflect the combination of tumor purity, total tumor copy number at a somatic SNV locus, and the number of tumor copies bearing the somatic allele. Clusters of somatic SNVs with similar allele depths inform the tumor model.

When a tumor has limited copy number variation and/or CNVs are mostly subclonal, such as in many liquid tumors, VAFs can help prevent incorrect or low-confidence estimated tumor models. Incorrect or low-confidence estimated tumor models can lead to wrong or filtered calls. VAF information can also help determine the presence or absence of a genome duplication even in clear, clonal CNVs.

To use VAF information, run somatic WGS CNV calling with small variant calling on tumor and matched-normal read and alignment inputs.

For example, you could use the following command line:

```
--enable-vc=true --enable-cnv=true --tumor-bam-input <Tumor_BAM> --bam-input
<Normal_BAM>
```

VAF-based modeling is enabled by default. To disable VAF-based modeling, set `--cnv-use-somatic-vc-vaf` to `false`.

HET-calling Mode

By default, DRAGEN uses HET-calling mode for segments with a copy number that is estimated to be heterogeneous (HET) among different subclones. Based on a statistical model, a segment is considered to be heterogeneous when the depths of BAF values in a segment are too far away from what is expected for the closest integer-copy number.

To turn on HET calling, specify `--cnv-somatic-enable-het-calling=true` on the command line.

When a segment is considered as heterogeneous, the output of the segment changes as follows.

- The HET tag is added to the INFO field for the segments.
- At least one of the CN and MCN values is given as non-REF values. Specifically, the values are given as the integer values closest to CNF and MCNF. If the integer values would result in a REF call, then at least one of the CN and MCN values is adjusted to the closest non-REF value.
- The ID, ALT, and GT fields are set appropriately for the chosen CN and MCN.
- The QUAL score reflects confidence that the segment has nonreference copy number in at least a fraction of the sample.
- The CNQ and MCNQ values reflect confidence that the assigned CN and MCN values are true in all of the samples, so at least one of the CNQ and MCNQ values are typically less than five.

 Even though DRAGEN reports CNF and MCNF for further inspection, CN and MCN should be used as the actual calls.

Somatic WGS CNV Model

Selecting a tumor purity and diploid coverage level (ploidy) is a key component of the somatic WGS CNV caller. The somatic WGS CNV caller uses a grid-search approach that evaluates many candidate models to attempt to fit the observed read counts and b-allele counts across all segments in the tumor sample. A log likelihood score is emitted for each candidate. The log likelihood scores are output to the `*.cnv.purity.coverage.models.tsv` file. The somatic WGS CNV caller chooses the purity, coverage pair with the highest log likelihood, and then computes several measures of model confidence based on the relative likelihood of the chosen model compared to alternative models.

If the confidence in the chosen model is low, the caller returns a default model with estimated tumor purity set to `NA`. The default model provides an alternative methodology to identify large somatic alterations (length of at least 1 Mb), and records are filtered based on their segment mean value (`SM`). The threshold values used by the caller are estimated automatically considering the variance of the sample, with larger SM thresholds for DUPS when the variance is higher. You can use alternative threshold values through the `--cnv-filter-del-mean` and `--cnv-filter-dup-mean` parameters. When the caller returns the default model, the fields regarding copy number states based on model estimation (ie, `CN`, `CNF`, `CNQ`, `MCN`, `MCNF`, and `MCNQ`) are omitted from the final VCF output, and HET-aware calling is enabled.

Grid search optimization informed by essential regions

To improve accuracy on the tumor ploidy model estimation, the somatic WGS CNV caller estimates whether the chosen model calls homozygous deletions on regions that are likely to reduce the overall fitness of cells, which are therefore deemed to be "essential" and under negative selection. In the current literature, recent efforts tried to map such cell-essential genes (eg, in 2015 - <https://www.science.org/doi/10.1126/science.aac7041>).

The check on essential regions is controlled with `--cnv-somatic-enable-lower-ploidy-limit` (default true). Default bedfiles describing the essential regions are provided for hg19, GRCh37, hs37d5, GRCh38, but a custom bedfile can also be provided in input through the `--cnv-somatic-essential-genes-bed=<BEDFILE_PATH>` parameter. In such case, the feature is automatically enabled. A custom essential regions bedfile needs to have the following format: 4-column, tab-separated, where the first 3 columns identify the coordinates of the essential region (chromosome, 0-based start, excluded end). The fourth column is the region id (string type). For the purpose of the algorithm, currently only the first 3 columns are used. However, the fourth might be helpful to investigate manually which regions drove the decisions on model plausibility made by the caller.

If the somatic WGS CNV caller does not find any overlap between any of the homozygous deletions and any of the essential regions, the model is considered plausible and the model optimization ends. Otherwise, when at least an overlap is found, the model is declared invalid and the model search is repeated on the subset of models that support at least one copy (CN = 1) for the essential region with the lowest coverage among the regions overlapping homozygous deletions.

Somatic WGS CNV Smoothing

The segmentation stage might produce adjacent or nearby segments that are assigned the same copy number and have similar depth and BAF data. This segmentation can result in a region with consistent true copy number being fragmented into several pieces. The fragmentation might be undesirable for downstream use of copy number estimates. Also, for some uses, it can be preferable to smooth short segments that would be assigned different copy numbers whether due to a true copy number change or an artifact. To reduce undesirable fragmentation, initial segments can be merged during a postcalling segment smoothing step.

After initial calling, segments shorter than the specified value of `--cnv-filter-length` are deemed negligible. Among the remaining nonnegligible segments, successive pairs are evaluated for merging. On a trial basis, the Somatic WGS CNV Caller combines two successive segments that are within `--cnv-merge-distance` of one another and have the same CN and MCN assignments, along with any intervening negligible segments into a single segment that is recalled and rescored. The default value is 10,000 for WGS Somatic CNV. If the merged segment receives the same CN and MCN as its constituent nonnegligible pieces with a sufficiently high-quality score, the original segments are replaced with the merged segment. The merged segment might be further merged with other initial or merged segments to either side. Merging proceeds until all segment pairs that meet the criteria are merged. NB. When the germline CN information is available, and two segments have different germline CN, they will not be merged.

Somatic WGS CNV VCF Output

The somatic WGS CNV VCF file follows the standard VCF format and contains the following differences from the germline CNV VCF output.

Headers

The following header lines are specific to somatic WGS CNV calling.

- ModelSource—The primary basis on which the final tumor model was chosen. The following values can be included.
 - DEPTH+BAF—Depth+BAF signal is used to determine tumor model.
 - DEPTH+BAF_DOUBLED—The initial depth+BAF model is duplicated based on VAF signal or excess segments at half the expected depth change.

- DEPTH+BAFDEDUPLICATED—The depth+BAF model is deduplicated based on VAF signal or insufficient segments supporting a duplication.
- DEPTH+BAFWEAK—Depth+BAF signal is used to determine lower-confidence tumor model.
- VAF—VAF signal is used to determine tumor model due to insufficient depth+BAF signal.
- DEGENERATE_DIPLOID—Sample is treated as high-purity diploid in absence of adequate signal from depth+BAF and VAF. The diploid coverage is set to lowest value observed in a substantial number of bases in segments with BAF=50%.
- SAMPLE_MEDIAN—Sample is treated as high-purity diploid in absence of adequate signal from depth+BAF and VAF. Diploid coverage set to sample median.
- EstimatedTumorPurity—Estimated fraction of cells in the sample due to tumor. The range of this field is [0, 1] or NA if a confident model could not be determined.
- DiploidCoverage—Expected read count for a target bin in a diploid region. The numeric value is unlimited.
- OverallPloidy—Length weighted average of tumor copy number for PASS events. The numeric value is unlimited.
- AlternativeModelDedup—An alternative to the best model corresponding to one less whole-genome duplication. The alternative is given as a pair of values (tumor purity, diploid coverage). This can be useful for manual investigation if the best model might involve a spurious genome duplication.
- AlternativeModelDup—An alternative to the best model corresponding to one more whole-genome duplication. The alternative is given as a pair of values (tumor purity, diploid coverage). This can be useful for manual investigation where the best model might have missed a true genome duplication.
- OutlierBafFraction—A QC metric that measures the fraction of b-allele frequencies that are incompatible with the segment the BAFs belong to. High values might indicate a mismatched-normal, substantial cross-sample contamination, or a different source of a mosaic genome, such as bone marrow transplantation. The range of this field is [0,1].

ID

The ID column represents the type of event. In addition to representing GAIN, LOSS, and REF events, the Copy Neutral Loss of Heterozygosity (CNLOH) and Copy Number Gain with LOH (GAINLOH) entries represent Loss of Heterozygosity (LOH) events.

ALT

The ALT field can contain two alleles, such as ,<DUP>, which allows representation of allele-specific copy numbers if they differ in copy number states.

FILTER

The following additional filters are applied to the FILTER field.

- binCount—Filters CNV events with a bin count lower than the threshold.
- segmentMean—Marks records as non-PASSING based on segment mean (SM) when the caller returns the default model. Segments having less than 1 Mb are automatically assigned this filter when returning the default model.

FORMAT

The FORMAT fields are described in the header section. The following fields are specific to somatic WGS CNV:

ID	Description
AS	Number of allelic read count sites
BC	Number of read count bins
CN	Estimated total copy number in tumor fraction of sample
CNF	Floating point estimate of tumor copy number
CNQ	Exact total copy number Q-score
MAF	Maximum estimate of the minor allele frequency
MCN	Estimated minor-haplotype copy number
MCNF	Floating point estimate of tumor minor-haplotype copy number
MCNQ	Minor copy number Q-score
NCN	Normal-sample copy number. The field is only present in germline-aware mode.
SCND	Difference between CN and GCN. The field is only present in germline-aware mode.
SD	Best estimate of bias-corrected read count for a segment
SM	Fold Change between segment corrected counts and baseline depth (ie, diploid for autosomal segments)

Allele Specific Copy Number Example

The Somatic WGS CNV Caller can report the total tumor copy number by estimating tumor purity. The BAF estimations from matched normal SNVs or population SNPs allow for allele specific copy number calling.

The following table provides examples for a DUP in a reference diploid region.

Total Copy Number	Minor Copy Number	ASCN Scenario
4	2	2+2
4	1	3+1
[LOH] 4*	0	4+0

*The entry represents a Loss of Heterozygosity (LOH) case. The total copy number is still considered a DUP, so the entry is annotated as GAINLOH to distinguish the value from Copy Neutral LOH (CNLOH), which would be annotated as 2+0.

Repeat Expansion Detection with ExpansionHunter

Short tandem repeats (STRs) are regions of the genome consisting of repetitions of short DNA segments called repeat units. STRs can expand to lengths beyond the normal range and cause mutations called repeat expansions. Repeat expansions are responsible for many diseases, including Fragile X syndrome, amyotrophic lateral sclerosis, and Huntington's disease.

DRAGEN includes a repeat expansion detection method called ExpansionHunter. ExpansionHunter performs sequence-graph based realignment of reads that originate inside and around each target repeat. ExpansionHunter then genotypes the length of the repeat in each allele based on these graph alignments.

The ExpansionHunter is designed for PCR-free whole genome samples. Repeats are only genotyped if the coverage at the locus is at least 10x. The ExpansionHunter cannot be run on multiple FASTQ files that are assigned to different library IDs in the `fastq_list.csv` file.

More information and analysis are available in the following ExpansionHunter papers:

- Dolzhenko et al., *Detection of long repeat expansions from PCR-free whole-genome sequence data* 2017
- Dolzhenko et al., *ExpansionHunter: A sequence-graph based tool to analyze variation in short tandem repeat regions* 2019

Repeat Expansion Detection Options

To enable DRAGEN repeat expansion detection, the following command line options are required.

- `--repeat-genotype-enable=true`
- `--repeat-genotype-specs=<path to specification file>`

You can use the `--sample-sex` option to specify the sex of the sample.

The following options are optional.

- `--repeat-genotype-region-extension-length=<length of region around repeat to examine> (default 1000 bp)`

- `--repeat-genotype-min-baseq=<Minimum base quality for high confidence bases>`
(default 20)

For more information on the specification file specified by `--repeat-genotype-specs` option, see [Repeat Expansion Specification Files](#).

The main output of repeat expansion detection is a VCF file that contains the variants found via this analysis.

Repeat Expansion Specification Files

The repeat-specification (also called variant catalog) JSON file defines the repeat regions for ExpansionHunter to analyze. Default repeat-specification for some pathogenic and polymorphic repeats are in the `/opt/edico/repeat-specs/` directory, based on the reference genome used with DRAGEN.

You can create specification files for new repeat regions by using one of the provided specification files as a template. See the ExpansionHunter documentation for details on the format.

`--repeat-genotype-specs` is required for ExpansionHunter. If the option is not provided, DRAGEN attempts to autodetect the applicable catalog file from `/opt/edico/repeat-specs/` based on the reference provided.

Users can choose between any of the three default repeat-specification files packaged with DRAGEN using the command line option: `--repeat-genotype-use-catalog=<default|default_plus_smn|expanded>`. The `default` option includes ~60 repeats. The `default_plus_smn` option includes the SMN repeat in addition to all the repeats in the default catalog. The expanded catalog includes ~50K repeats, see [Covered Repeat Regions](#). If `--repeat-genotype-use-catalog` is not specified on the command line, then the default catalog is used.

The repeat genotyping results will be incorrect if the selected reference genome is not compatible with the repeat specification file. When this occurs, many repeats may be marked as `LowDepth` in the VCF output file or estimated to have zero length. This can be further confirmed by visualizing read alignments with the REViewer visualization tool available on github.

Covered Repeat Regions

The default variant catalog contains specifications on disease-causing repeats located in AFF2, AR, ARX_1, ARX_2, ATN1, ATXN1, ATXN10, ATXN2, ATXN3, ATXN7, ATXN8OS, BEAN1, C9ORF72, CACNA1A, CBL, CNBP, COMP, CSTB, DAB1, DIP2B, DMD, DMPK, EIF4A3, FMR1, FOXL2, FXN, GIPC1, GLS, HOXA13_1, HOXA13_2, HOXA13_3, HOXD13, HTT, JPH3, LRP12, MARCHF6, NIPA1, NOP56, NOTCH2NLC, NUTM2B-AS1, PABPN1, PHOX2B, PPP2R2B, PRDM12, PRNP, RAPGEF2, RFC1, RUNX2, SAMD12, SOX3, STARD7, TBP, TBX1, TCF4, TNRC6A, VWA1, XYLT1, YEATS2, ZIC2 and ZIC3 genes.

For the expanded variant catalog, apart from the aforementioned disease-causing repeats, there are ~50K additional polymorphic repeats. They are initially detected using STR-Finder from the 1000 Genomes Project. After that, the candidate repeats are filtered out based on a customized quality control pipeline. Finally, the top 50K repeats which locate closest to exon regions, are included in the final catalog.

The ExpansionHunter can detect pathogenic expansions of FXN, ATXN3, ATN1, AR, DMPK, HTT, FMR1, ATXN1, C9ORF72 repeats with high accuracy (see the ExpansionHunter papers above). The pathogenicity status of some repeats might depend on the presence of sequence interruptions or motif changes that ExpansionHunter does not call. If you would like to visually inspect the relevant read alignments, you can use a Repeat Expansion Viewer third-party tool.

Repeat Expansion Detection Output Files

VCF Output File

The results of repeat genotyping are output as a separate VCF file, which provides the length of each allele at each callable repeat defined in the repeat-specification catalog file. The name is <outputPrefix>.repeats.vcf (*.gz).

The VCF output file lists with the following fields first.

Table 4 Core VCF Fields

Field	Description
CHROM	Chromosome identifier
POS	Position of the first base before the repeat region in the reference
ID	Always .
REF	The reference base at position POS
ALT	List of repeat alleles in format <STRn> . N is the number of repeat units.
QUAL	Always .
FILTER	LowDepth filter is applied when the overall locus depth is below 10x or the number of reads that span one or both breakends is below 5.

Table 5 Additional INFO Fields

Field	Description
END	Position of the last base of the repeat region in the reference
REF	Number of repeat units spanned by the repeat in the reference
RL	Reference length in bp

Field	Description
VARID	Variant ID from the variant catalog
RU	Repeat unit in the reference orientation
REPID	Variant ID from the variant catalog

Table 6 GENOTYPE (Per Sample) Fields

Field	Description
GT	Genotype
SO	Type of reads that support the allele. Values can be SPANNING, FLANKING, or INREPEAT. These values indicate if the reads span, flank, or are fully contained in the repeat.
REPCN	Number of repeat units spanned by the allele
REPCI	Confidence interval for REPCN
ADSP	Number of spanning reads consistent with the allele
ADFL	Number of flanking reads consistent with the allele
ADIR	Number of in-repeat reads consistent with the allele
LC	Locus Coverage

For example, the following VCF entry describes the ATXN1 repeat in a sample NA13537.

```
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT NA13537
chr6 16327864 . G <STR33>,<STR58> . PASS
END=16327954;REF=30;RL=90;RU=TGC;VARID=ATXN1;REPID=ATXN1
GT:SO:REPCN:REPCI:ADSP:ADFL:ADIR:LC 1/2:SPANNING/INREPEAT:33/58:33-33/52-
71:4/0:69/83:0/4:37.459459
```

In this example, the first allele spans 33 repeat units while the second allele spans 58 repeat units. The repeat unit is TGC (RU INFO field), so the sequence of the first allele is TGC x 33 and the sequence of the second allele is TGC x 58. The repeat spans 30 repeat units in the reference (REF INFO field).

The length of the short allele was estimated from spanning reads (SPANNING) while the length of the expanded allele was estimated from in-repeat reads (INREPEAT). The confidence interval for the size of the expanded allele is (52,71). There are 4 spanning and 69 flanking reads consistent with the repeat allele of size 33 that is 4 reads fully contain the repeat of size 33 and 69 flanking reads overlap at most 33 repeat units. There are 83 flanking and 4 in-repeat reads consistent with the repeat allele of size 58. The average coverage of this locus is 37.46x.

Additional Output Files

The sequence-graph alignments of reads in the targeted repeat regions are output in a BAM file. You can use a specialized GraphAlignmentViewer tool available on GitHub to visualize the alignments. Programs like Integrative Genomics Viewer (IGV) are not designed for displaying graph-aligned reads and cannot visualize these BAMs.

The BAMs store graph alignments in custom XG tags using the format

<LocusName>, <StartPosition>, <GraphCIGAR>.

- LocusName—A locus identifier that matches the corresponding entry in the repeat expansion specification file.
- StartPosition—The starting alignment position of a read on the first graph node.
- GraphCIGAR—The alignment of a read against the graph starting from that position. GraphCIGAR consists of a sequence of graph node identifiers and linear CIGARS describing the alignment of the read to each node.

Quality scores in the BAM file are binary. High-scoring bases are assigned a score of 40, and low-scoring bases are assigned a score of 0.

Paralog Calling

Targeted Variant Calling

Repetitive regions in the human genome pose a challenge for general variant calling approaches which typically cannot make use of any misplaced MAPQ0 reads. Furthermore, high sequence homology of some genes with a pseudogene paralog can lead to a wide variety of common structural variants (SVs) in the population, requiring specialized targeted calling approaches. DRAGEN supports targeted calling for a number of genes/targets as described in subsequent target-specific sections.

Each targeted caller can be enabled/disabled separately (refer to target-specific section), or all targeted callers can be enabled using the command line option --enable-targeted=true. The targeted callers produce a <prefix>.targeted.json containing a gene-level summary of the variant caller results. Additionally, the details of individual variant calls are reported in a <prefix>.targeted.vcf.gz.

Input Data

All targeted callers require WGS data aligned to a human reference genome with at least 30x coverage. Human reference genome builds based on hg19, GRCh37, and hg38 are supported.

Output Files

Targeted JSON File

The targeted callers generate a <output-file-prefix>.targeted.json file in the output directory. The output file is a JSON formatted file containing the fields below.

Fields in JSON	Explanation	Type and Possible Values	Present
sample	The sample name.	string	always
dragenVersion	The version of DRAGEN.	string	always
pharmcatMetabolismStatusResourceUrl	Web URL for the PharmCAT resource used for calling metabolism status.	string	PGx gene target is enabled
cyp2b6	The CYP2B6 targeted caller specific fields.	dictionary	CYP2B6 caller is enabled
cyp2d6	The CYP2D6 targeted caller specific fields.	dictionary	CYP2D6 caller is enabled
cyp21a2	The CYP21A2 targeted caller specific fields.	dictionary	CYP21A2 caller is enabled
gba	The GBA targeted caller specific fields.	dictionary	GBA caller is enabled
hba	The HBA targeted caller specific fields.	dictionary	HBA caller is enabled
lpa	The LPA targeted caller specific fields.	dictionary	LPA caller is enabled
rh	The RH targeted caller specific fields.	dictionary	RH caller is enabled
smn	The SMN targeted caller specific fields.	dictionary	SMN caller is enabled
star_allele	The Star Allele specific fields.	dictionary	Star Allele is enabled

Targeted VCF File

The targeted callers generate a `<output-file-prefix>.targeted.vcf[.gz]` file in the output directory. The output file is a VCFv4.2 formatted file possibly compressed. The targeted callers that support VCF output are: RH, HBA, LPA, and SMN.

Small variant calls and copy number variant calls are reported in the same VCF file.

The `<output-file-prefix>.targeted.vcf[.gz]` file includes the source header line defining the source of the VCF calls as from `DRAGEN_TARGETED`.

```
##source=DRAGEN_TARGETED
```

The main INFO fields that are used in the `<output-file-prefix>.targeted.vcf[.gz]` file are the `EVENT` and `EVENTTYPE` INFO fields.

The `EVENT` and `EVENTTYPE` INFO fields are formally introduced in VCFv4.4 to enable the representation of complex rearrangements. This is achieved using the `EVENT` field to group all the related VCF records together, and the `EVENTTYPE` to classify the event. The corresponding header lines are the following.

```
##INFO=<ID=EVENT,Number=A>Type=String,Description="Event name">
##INFO=<ID=EVENTTYPE,Number=A>Type=String,Description="Type of associated
event">
```

However, the use of `EVENT` is not limited to complex rearrangements and can be used to associate non-symbolic alleles, for example in cases of variant position ambiguity in high homology regions.

Since the `EVENTTYPE` values are implementation-defined, custom `EVENTTYPE` header lines are included to describe each `EVENTTYPE`.

```
##EVENTTYPE=<ID=GENE_CONVERSION,Description="Gene conversion event">
##EVENTTYPE=<ID=VARIANT_IN_HOMOLOGY_REGION,Description="Variant in homology
region">
##EVENTTYPE=<ID=VNTR,Description="Variable number tandem repeat">
```

Variants In High Homology Regions

In the case of target variants in a high homology region, the variant is reported in all homologous regions with a ploidy corresponding to the total copy number of all the homologous regions.

In the depicted example there are two genes A and B that include a high homology region. The usual process to call variants in this regions is to make a joint pileup of the reads aligning in both genes A and B and call the variants using a model with a ploidy proportional to the total copy number of the regions. This generates divergent possible genotypes that are equally likely since the variant cannot be confidently placed in either gene A or gene B. The variant is reported as follows:

```
chr1 100 . A T . TargetedRepeatConflict EVENT=GeneA-B:50A>T;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION GT 0/0/0/1
chr1 200 . A T . TargetedRepeatConflict EVENT=GeneA-B:50A>T;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION GT 0/0/0/1
```

Given the unconventional ploidy representation of variants in high homology regions, a TargetedRepeatConflict filter is applied to the records. The header line for the filter is the following.

```
##FILTER=<ID=TargetedRepeatConflict,Description="Set if call conflicts with a targeted call">
```

Gene Conversion Events

In the case of an identified gene conversion the differentiating sites as variants in the acceptor region are reported.

In the depicted example there are two genes A and B and gene A is the acceptor of a gene conversion from gene B (green box in the figure). Gene conversion are identified by observing variations in copy number at differentiating sites (blue and pink bars in the figure) in consecutive regions. Copy number variations between regions define the breakends of the gene conversion. There are two possible way to report the gene conversion:

1. Reporting the differentiating sites variants in the acceptor gene as SNV entries.
2. Report a variation in copy number and gene conversion break ends as CNV and SV entries.

Only the small variant representation is currently supported, and is as follows.

```
chr1 121 . A T . PASS EVENT=GC_AB;EVENTTYPE=GENE_CONVERSION; GT:PS 0|1:121
...
chr1 280 . G A . PASS EVENT=GC_AB;EVENTTYPE=GENE_CONVERSION; GT:PS 0|1:121
```

In the case of an observed gene conversion event, there may be differentiating sites with a genotype that is inconsistent with that gene conversion event. In these cases the RecombinantConflict filter is applied. The RecombinantConflict is defined by the following header line.

```
##FILTER=<ID=RecombinantConflict,Description="Set if call has a copy number that conflicts with a recombinant variant">
```

In the example, the resulting representation is as follows:

```
chr1 121 . A T . PASS EVENT=GC_AB;EVENTTYPE=GENE_CONVERSION; GT:PS 0|1:121
...
```

```
chr1 144 . C T . RecombinantConflict EVENT=GC_AB;EVENTTYPE=GENE_CONVERSION;
GT:PS 1|1:121
chr1 153 . A G . RecombinantConflict EVENT=GC_AB;EVENTTYPE=GENE_CONVERSION; GT
0/0
...
chr1 280 . G A . PASS EVENT=GC_AB;EVENTTYPE=GENE_CONVERSION; GT:PS 0|1:121
```

Reference Copy Number Representation

The use of `GT=0` for symbolic structural variant alleles is formally disambiguated in VCFv4.4, specifying that "`GT=0` indicates the absence of any of the `ALT` symbolic structural variants defined in the record".

In the depicted example there are two regions, one with a heterozygous DUP followed by a REF copy number region. The CNV representation is as follows:

```
chr1 100 . A <DUP> . . END=300;EVENT=A;SVCLAIM=D;SVLEN=200;CN=2 GT:CN 0/1:3
chr1 300 . A <CNV> . . END=500;EVENT=A;SVCLAIM=D;SVLEN=200; GT:CN 0/0:2
```

The relevant header lines for the VCF records above are the following:

```
##INFO=<ID=CN,Number=A,Type=Float,Description="Copy number of CNV /
breakpoint">
##INFO=<ID=END,Number=1,Type=Integer,Description="End position of the variant
described in this record">
##INFO=<ID=SVLEN,Number=A,Type=Integer,Description="Difference in length
between REF and ALT alleles">
##INFO=<ID=SVCLAIM,Number=A,Type=String,Description="Claim made by the
structural variant call. Valid values are D, J, DJ for abundance, adjacency and
both respectively.">
##FORMAT=<ID=CN,Number=1,Type=Float,Description="Estimated copy number">
```

Overlapping Variants Representation

In the depicted example there are two overlapping CNVs that can be represented as follows:

```
chr1 100 . A <DUP> . . END=300;EVENT=A;SVCLAIM=D;SVLEN=200;CN=2 GT:PS .|1:100
chr1 200 . G <DEL> . . END=500;EVENT=A;SVCLAIM=D;SVLEN=300;CN=0 GT:PS 1|.:100
```

Variable Number Tandem Repeat Representation

In the depicted example there is a Variable Number Tandem Repeat (VNTR) region composed of three repeat units in the reference. The `CN` INFO field is used to report the allele copy number, the `CN` FORMAT field to is used report the region total copy number given by the sum of the allele copy numbers, and the `REPCN` FORMAT field is used to report the repeat unit copy number equal to the allele copy number multiplied by the number of repeat units in the reference.

This VNTR can be represented as follows:

```
chr1 100 . A <DUP>,<DUP> . .
END=400;EVENT=A;EVENTTYPE=VNTR;SVCLAIM=D;SVLEN=300;CN=2.6,4.3 GT:CN:REPCN
1|2:6.9:8|13
```

The `REPCN` header line is the following.

```
##FORMAT=<ID=REPCN,Number=1,Type=String,Description="Number of repeat units
spanned by the allele">
```

Additional Filters

The `TargetedLowQual` filter is applied if the `QUAL` of a targeted variant is less than 3.00.

```
##FORMAT=<ID=REPCN,Number=1,Type=String,Description="Number of repeat units
spanned by the allele">
```

The `TargetedLowGQ` filter is applied if the targeted variant has `GQ` smaller than 3.

```
##FILTER=<ID=TargetedLowGQ,Description="Set if call has GQ < 3">
```

Merging Targeted Calls In The hard-filtered Files

The small variant target specific VCF calls can be merged into the `<prefix>.hard-filtered.vcf.gz` and `<prefix>.hard-filtered.gvcf.gz` files, briefly `hard-filtered files`, using the `--targeted-merge-vc` flag by providing the caller name(s). For example, `--targeted-merge-vc rh` will enable merging of the calls from the `rh` caller into the `hard-filtered files` and `--targeted-merge-vc rh hba` will enable merging of the calls from the `rh` and `hba` callers into the `hard-filtered files`. The `all` value will merge all calls from all supported targeted callers into the `hard-filtered files`, while the `none` value will merge no calls into the `hard-filtered files`.

The default value for `--targeted-merge-vc` is `rh`, therefore the calls from the `rh` caller are merged into the `hard-filtered files` by default.

The targeted calls merged into the `hard-filtered files` are marked with a `TARGETED` INFO flag.

When enabled, targeted small variants are merged into the hard-filtered files regardless of any regions that may be provided using the `--vc-target-bed` option.

Merging Strategy

The merging strategy for targeted small variant calls is to prioritize the targeted calls over small variant calls from the germline small variant caller. When a germline small variant call overlaps a targeted caller call, then the small variant call is filtered with a `TargetedConflict` filter if any of the following holds:

- The targeted caller call is `PASS`.
- The small variant call and targeted caller call have incompatible genotypes and the targeted caller call is not filtered with the `TargetedLowGQ` filter.

The strategy is summarized in the following examples.

1. The TARGETED call is `PASS`

```
chr1 100 . A C . TargetedConflict . GT 0/1
chr1 100 . A C . PASS TARGETED GT 1/1
```

2. The TARGETED call and the small variant call are not overlapping

```
chr1 110 . T TCA . PASS . GT 0/1
chr1 111 . G A . PASS TARGETED GT 0/1
```

3. The TARGETED call is filtered with `TargetedLowQual` and has a discordant variant representation with the overlapping small variant call

```
chr1 120 . ATTC A . TargetedConflict . GT 0/1
chr1 121 . T A . TargetedLowQual TARGETED GT 0/1
chr1 125 . TCAC T . TargetedLowQual TARGETED GT 0/1
chr1 126 . C G . TargetedConflict . GT 0/1
```

4. The TARGETED call is filtered with `TargetedLowQual` and has a discordant genotype with the overlapping small variant call

```
chr1 130 . C G . TargetedConflict . GT 0/1
```

```
chr1 130 . C     G     . TargetedLowQual    TARGETED    GT 1/1
```

5. The TARGETED call is filtered with TargetedLowGQ and has a discordant genotype with the overlapping small variant call

```
chr1 140 . AC    A     . PASS           .          GT:GQ 0/1:5
```

```
chr1 140 . A     T     . TargetedLowGQ   TARGETED   GT:GQ 1/1:2
```

CYP2B6 Caller

The CYP2B6 Caller is capable of genotyping the CYP2B6 gene from whole-genome sequencing (WGS) data. Due to high sequence similarity with its pseudogene paralog CYP2B7 and a wide variety of common structural variants (SVs), a specialized caller is necessary to resolve variants and identify likely star allele haplotypes.

The CYP2B6 Caller performs the following steps:

1. Determines total CYP2B6 and CYP2B7 copy number from read depth.
2. Determines CYP2B6-derived copy number at CYP2B6/CYP2B7 differentiating sites.
3. Detects SV breakpoints by calculating the changes in CYP2B6-derived copy number along the CYP2B6 gene.
4. Calls small variants in CYP2B6 copies.
5. Identifies star alleles from the detected SV breakpoints and small variants.
6. Identifies the most likely genotype for the called star alleles.
7. The CYP2B6 Caller requires WGS data aligned to a human reference genome with at least 30x coverage.

Total CYP2B6 and CYP2B7 Copy Number

The first step of CYP2B6 calling is to determine the combined copy number of CYP2B6 and CYP2B7. Reads aligned to regions in either CYP2B6 or CYP2B7 are counted. The counts in each region are corrected for GC-bias, and then normalized to a diploid baseline. The GC-bias correction and normalization factors are determined from read counts in 3000 preselected 2 kb regions across the genome. These 3000 normalization regions were randomly selected from the portion of the reference genome having stable coverage across population samples. The combined CYP2B6 and CYP2B7 copy number is then calculated from the average sequencing depth across the CYP2B6 and CYP2B7 regions.

Differentiating Sites

The CYP2B6-derived copy number is calculated at 99 predefined differentiating sites across the CYP2B6 gene. The differentiating sites are selected at positions with sequence differences in CYP2B6 and CYP2B7 where calling the CYP2B6-derived copy number shows an accuracy of greater than 98% based on sequencing data from the 1000 Genomes Project.

For each differentiating site, CYP2B6-specific and CYP2B7-specific alleles are counted in reads mapping to either CYP2B6 or the homologous region in CYP2B7. The CYP2B6-derived copy number is then calculated from the two gene-specific allele counts using the total CYP2B6 and CYP2B7 copy number calculated from the previous step.

Structural Variant Calling

The CYP2B6-derived copy number along the CYP2B6 gene is used to identify known population structural variants (SVs), including whole gene deletions and duplications as well as certain gene conversions and gene fusions. The following fusion variants are detected:

Fusion Breakpoint	Hybrid Gene Structure	Star-Allele Designation
intron 4-exon 5	2B7-2B6	*29
intron 4-exon 5	2B6-2B7	*30

Small Variant Calling

35 small variants that define various star alleles are detected from the read alignments. All of these variants are in unique (nonhomologous) regions of CYP2B6 with high mapping quality. Only reads mapping to CYP2B6 are used for calling variants in nonhomologous regions.

For each variant, reads containing either the variant allele or the nonvariant alleles are counted. A binomial model that incorporates the sequencing errors is then used to determine the most likely variant copy number (0 for nonvariant).

Samples with poor sequencing quality or greater than five copies of CYP2B6 will have allele counts with higher variance. This elevated variance increases the chance that the most likely variant copy number is wrong. To handle these cases, the small variant caller also indicates alternate, less likely variant copy numbers.

Recombinant Variant Calling

The recombinant (gene conversion) variant 18053A>G is detected by phasing the variant site with five flanking differentiating sites. When the haplotypes formed from phasing these sites supports the gene conversion in CYP2B6, a read depth analysis at the gene conversion breakpoints (transitions from either CYP2B6->CYP2B7 or CYP2B7->CYP2B6) is performed. Given the posterior probability that there is at least one gene conversion variant is above 0.7 then DRAGEN uses the variant for star allele identification.

Star Allele Identification

The called SVs and small variant genotypes are matched against the definitions of 39 different star alleles. This might result in different sets of star alleles matching the called variant genotypes, such as with *1, *6 and *4, *49 where both sets of star alleles contain the same two small variants. When the small variant caller emits alternate, less likely variant copy numbers in addition to the most likely variant copy numbers this might result in different sets of star alleles being identified, because these alternate sets of variant copy numbers are also matched to the star allele definitions. The number of matched star alleles must match the number of CYP2B6-derived gene copies determined from previous steps. If no variant genotypes can be matched to a set of star alleles, the CYP2B6 Caller returns a no call during the genotyping step with filter value `No_call`.

Genotyping

Given a possible set of star alleles, the genotyping step attempts to identify the two likely haplotypes that contain all star alleles in the set. The likelihood of any given genotype is determined from a table of population frequencies determined from the 1000 Genomes Project and the genotype with the highest population frequency is selected. When two or more possible genotypes are identified with similar population frequencies, then all genotypes are emitted. This results in a call with filter value `More_than_one_possible_genotype`.

CYP2B6 Output File

The CYP2B6 Caller prints out its calls in the targeted callers output file, `<prefix>.targeted.json` (that also aggregates calls from other targeted callers). An example of this file with the CYP2B6 caller set is as follows:

```
{
  "dragenVersion": "4.2.0-724-gb600fce",
  "sample": "NA19374",
  "pharmacatMetabolismStatusResourceUrl": "https://github.com/PharmGKB/PharmCAT/blob/aeecfe5f787e95dfb31ede62884e287affef45b3/src/main/resources/org/pharmgkb/pharmacat/definition/gene_phenotypes.json",
  "cyp2b6": {
    "genotype": "*17/*2",
    "genotypeFilter": "PASS",
    "pharmacatDescription": "An individual carrying two normal function alleles",
  }
}
```

```

    "pharmcatMetabolismStatus": "Normal Metabolizer"
},
}

```

For CYP2B6 caller, the fields are defined as follows.

Fields in JSON	Explanation	Type and Possible Values
dragenVersion	Version of DRAGEN	string
sample	sample id	string
pharmcatMetabolismStatusResourceUrl	an URL containing the genotype to PharmCAT mapping information	string (web link)
cyp2b6	a json array containing the CYP2B6 call for this sample	json-array
genotype	star allele genotype identified for sample	string
genotypeFilter	The filter status for the genotype call	string (The value can include: PASS, No_call, or More_than_one_possible_genotype)

Fields in JSON	Explanation	Type and Possible Values
pharmacatDescription	The description corresponding to the genotype, mapped from PharmCAT	string
pharmacatMetabolismStatus	The metabolism status corresponding to the genotype, mapped from PharmCAT	string

When the option `--targeted-enable-legacy-output=true` is set, the CYP2B6 Caller also generates a `<output-file-prefix>.cyp2b6.tsv` file in the output directory. The output file contains a single line containing the tab-delimited fields below. The output file contains no header line.

- Sample name
- One or more semicolon-delimited CYP2B6 genotypes or None for no call
- The filter status. The value can include: PASS, No_call, or More_than_one_possible_genotype

Each CYP2B6 genotype contains two haplotypes separated by a slash (eg `*1/*2`). Each haplotype consists of one or more star alleles separated by a plus sign (eg `*10+*36`). When a haplotype contains more than one copy of the same star allele, that star allele only appears once and is followed by a multiplication sign, and then the number of copies (eg `*1x2` for two copies of `*1`).

Output File Examples

The following are example output files:

```
HG00554 *17/*6 PASS
```

```
HG02621 *6/*U1 PASS
```

```
NA20348 *18/*6;*18/*9 More_than_one_possible_genotype
```

```
NA19003 None No_call
```

Command-line Examples

To enable the CYP2B6 Caller, use `--enable-cyp2b6=true`. The CYP2B6 Caller is disabled by default. The CYP2B6 Caller can run directly from FASTQ input with the mapper or from prealigned BAM/CRAM input. You can also enable the CYP2B6 Caller in parallel with any other germline variant callers as part of a WGS germline analysis workflow. For more information on other variant callers, see DNA Pipeline for DRAGEN.

FASTQ Input

The following command-line example uses FASTQ input:

```
dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/8 \
--fastq-file1 /staging/test/data/NA12878_R1.fastq \
--fastq-file2 /staging/test/data/NA12878_R2.fastq \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--RGID DRAGEN_RGID \
--RGSM NA12878 \
--enable-map-align=true \
--enable-cyp2b6=true
```

Prealigned BAM Input

The following command-line example uses BAM input that has already been aligned:

```
dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/8 \
--bam-input /staging/test/data/NA12878.bam \
--output-directory /staging/test/output \
```

```
--output-file-prefix NA12878_dragen \
--enable-map-align=false \
--enable-cyp2b6=true
```

CYP2D6 Caller

The CYP2D6 Caller is capable of genotyping the CYP2D6 gene from whole-genome sequencing (WGS) data and is derived from the method implemented in Cyrius¹. Due to high sequence similarity with its pseudogene paralog CYP2D7 and a wide variety of common structural variants (SVs), a specialized caller is necessary to resolve variants and identify likely star allele haplotypes.

The CYP2D6 Caller performs the following steps:

1. Determines total CYP2D6 and CYP2D7 copy number from read depth.
2. Determines CYP2D6-derived copy number at CYP2D6/CYP2D7 differentiating sites.
3. Detects SV breakpoints by calculating the changes in CYP2D6-derived copy number along the CYP2D6 gene.
4. Calls small variants in CYP2D6 copies.
5. Identifies star alleles from the detected SV breakpoints and small variants.
6. Identifies the most likely genotype for the called star alleles.

The CYP2D6 Caller requires WGS data aligned to a human reference genome with at least 30x coverage.

Total CYP2D6 and CYP2D7 Copy Number

The first step of CYP2D6 calling is to determine the combined copy number of CYP2D6 and CYP2D7. Reads aligned to regions in either CYP2D6 or CYP2D7 are counted. The counts in each region are corrected for GC-bias, and then normalized to a diploid baseline. The GC-bias correction and normalization factors are determined from read counts in 3000 preselected 2 kb regions across the genome. The combined CYP2D6 and CYP2D7 copy number is then calculated from the average sequencing depth across the CYP2D6 and CYP2D7 regions.

Differentiating Sites

The CYP2D6-derived copy number is calculated at 117 predefined differentiating sites across the CYP2D6 gene. The differentiating sites are selected at positions with sequence differences in CYP2D6 and CYP2D7 where calling the CYP2D6-derived copy number shows an accuracy of greater than 98% based on sequencing data from the 1000 Genomes Project.

For each differentiating site, CYP2D6-specific and CYP2D7-specific alleles are counted in reads mapping to either CYP2D6 or the homologous region in CYP2D7. The CYP2D6-derived copy number is then calculated from the two gene-specific allele counts using the total CYP2D6 and CYP2D7 copy number calculated from the previous step.

Structural Variant Calling

The CYP2D6-derived copy number along the CYP2D6 gene is used to identify known population structural variants (SVs), including whole gene deletions and duplications as well as certain gene conversions and gene fusions. The following fusion variants are detected:

Fusion Breakpoint	Hybrid Gene Structure	Star-Allele Designation
exon 9	2D6-2D7	*4.013, *36, *57, *83
exon 9	2D7-2D6	*13
intron 4	2D7-2D6	*13
intron 1	2D7-2D6	*13
intron 1	2D6-2D7	*68

In addition to the exon 9 fusion breakpoints, exon 9 can participate in CYP2D7 gene conversion resulting in an embedded CYP2D7 sequence instead of a true hybrid. The structural variant caller also detects exon 9 gene conversions. Because only changes in CYP2D6-derived copy number yield structural variant calls, there might be rare cases where two hybrid copies result in no structural variant calls. For example, when both *36 and *13 with fusion breakpoint in exon 9 are present. However, the structural variant caller is capable of detecting multiple copies of the same fusion type (eg, *36x2) or cases where both an exon 9 gene conversion copy and an exon 9 2D6-2D7 hybrid are present.

Small Variant Calling

118 small variants that define various star alleles are detected from the read alignments. 96 of these variants are in unique (nonhomologous) regions of CYP2D6 with high mapping quality. Only reads mapping to CYP2D6 are used for calling variants in nonhomologous regions. The other 22 variants occur in homologous regions of CYP2D6 where reads mapping to either CYP2D6 or CYP2D7 are used for variant calling.

For each variant, reads containing either the variant allele or the nonvariant alleles are counted. A binomial model that incorporates the sequencing errors is then used to determine the most likely variant copy number (0 for nonvariant). A strand bias filter is applied for certain noisy variants that tend to have false positive calls.

Samples with poor sequencing quality or greater than five copies of CYP2D6 will have allele counts with higher variance. This elevated variance increases the chance that the most likely variant copy number is wrong. To handle these cases, the small variant caller also indicates alternate, less likely variant copy numbers.

Star Allele Identification

The called SVs and small variant genotypes are matched against the definitions of 128 different star alleles. This might result in different sets of star alleles matching the called variant genotypes, such as with *1, *46 and *43, *45 where both sets of star alleles contain the same 4 small variants. When the small variant caller emits alternate, less likely variant copy numbers in addition to the most likely variant copy numbers this might result in different sets of star alleles being identified, since these alternate sets of variant copy numbers are also matched to the star allele definitions. The number of matched star alleles must match the number of CYP2D6-derived gene copies determined from previous steps. When there are fewer than two CYP2D6-derived gene copies, then one or more *5 deletion haplotypes are included in the output set of star alleles. If all variant genotypes cannot be matched to a set of star alleles, the CYP2D6 Caller returns a no call during the genotyping step with filter value `No_call`.

Genotyping

Given a possible set of star alleles, the genotyping step attempts to identify the two likely haplotypes that contain all star alleles in the set. The deletion haplotype (*5) is considered as a possible haplotype during this process. The likelihood of any given genotype is determined from a table of population frequencies determined from the 1000 Genomes Project and the genotype with the highest population frequency is selected. When two or more possible genotypes are identified with similar population frequencies, then all genotypes are emitted. This results in a call with filter value `More_than_one_possible_genotype`.

CYP2D6 Output File

The CYP2D6 Caller prints out its calls in the targeted callers output file, `<prefix>.targeted.json` (that also aggregates calls from other targeted callers). An example of this file with the CYP2D6 caller set is as follows:

```
{
  "dragenVersion": "4.2.0-724-gb600fcef",
  "sample": "NA19374",
```

```

    "pharmcatMetabolismStatusResourceUrl":  

    "https://github.com/PharmGKB/PharmCAT/blob/aeecfe5f787e95dfb31ede62884e287affef45b3/src/main/resources/org/pharmgkb/pharmcat/definition/gene_phenotypes.json",  

    "cyp2d6": {  

        "genotype": "*17/*2",  

        "genotypeFilter": "PASS",  

        "pharmcatDescription": "An individual carrying two normal function alleles",  

        "pharmcatMetabolismStatus": "Normal Metabolizer"  

    },  

}

```

For CYP2D6 caller, the fields are defined as follows.

Fields in JSON	Explanation	Type and Possible Values
dragenVersion	Version of DRAGEN	string
sample	sample id	string
pharmcatMetabolismStatusResourceUrl	an URL containing the genotype to PharmCAT mapping information	string (web link)
cyp2d6	a json array containing the CYP2D6 call for this sample	json-array
genotype	star allele genotype identified for sample	string
genotypeFilter	The filter status for the genotype call	string (The value can include: PASS, No_call, or More_than_one_possible_genotype)

Fields in JSON	Explanation	Type and Possible Values
pharmacatDescription	The description corresponding to the genotype, mapped from PharmCAT	string
pharmacatMetabolismStatus	The metabolism status corresponding to the genotype, mapped from PharmCAT	string

When the option `--targeted-enable-legacy-output=true` is set, the CYP2D6 Caller also generates a `<output-file-prefix>.cyp2d6.tsv` file in the output directory. The output file contains the following tab-delimited fields with no header line:

- Sample name.
- One or more semicolon-delimited CYP2D6 genotypes or `None` for no call.
- The filter status. The value can include: `PASS`, `No_call`, or `More_than_one_possible_genotype`.

Each CYP2D6 genotype contains two haplotypes separated by a slash (eg, `*1/*2`). Each haplotype consists of one or more star alleles separated by a plus sign (eg, `*10+*36`). When a haplotype contains more than one copy of the same star allele, that star allele only appears once and is followed by a multiplication sign, and then the number of copies (eg, `*1x2` for two copies of `*1`).

Output File Examples

The following are example output files:

```
NA18632 *10+*36x2/*52 PASS
HG01190 *4+*68/*5 PASS
NA17244 *2/*4x2+*13+*83;*2/*4+*4.013+*13+*39 More_than_one_possible_genotype
NA19908 *1/*46;*43/*45 More_than_one_possible_genotype
NA18611 None No_call
```

Command-line Examples

To enable the CYP2D6 Caller, use `--enable-cyp2d6=true`. The CYP2D6 Caller is disabled by default. The CYP2D6 Caller can run directly from FASTQ input with the mapper or from prealigned BAM/CRAM input. You can also enable the CYP2D6 Caller in parallel with any other germline variant callers as part of a WGS germline analysis workflow. For more information on other variant callers, see [DRAGEN DNA Pipeline on page 79](#).

FASTQ Input

The following command-line example uses FASTQ input:

```
dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/8 \
--fastq-file1 /staging/test/data/NA12878_R1.fastq \
--fastq-file2 /staging/test/data/NA12878_R2.fastq \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--RGID DRAGEN_RGID \
--RGSM NA12878 \
--enable-map-align=true \
--enable-cyp2d6=true
```

Prealigned BAM Input

The following command-line example uses BAM input that has already been aligned:

```
dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/8 \
--bam-input /staging/test/data/NA12878.bam \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--enable-map-align=false \
--enable-cyp2d6=true
```

¹ ChenX, Shen F, Gonzaludo N, et al. Cyrius: accurate CYP2D6 genotyping using whole-genome sequencing data. *The Pharmacogenomics Journal*. 2021;21(2):251-261. doi:10.1038/s41397-020-00205-5

CYP21A2 Caller

The CYP21A2 Caller is capable of genotyping the CYP21A2 gene from whole-genome sequencing (WGS) data. Due to high sequence similarity with its pseudogene paralog CYP21A1P and a wide variety of common structural variants (SVs), a specialized caller is necessary to resolve variants.

The CYP21A2 calling workflow is broken up into the following major stages:

1. Loading input configuration
2. Processing read data
3. Analyzing read data

Read data analysis is further split into the following steps:

1. Determine total CYP21A2 and CYP21A1P copy number from read depth.
2. Call small variants in CYP21A2 copies.
3. Phase reads to detect common variants and recombination events.
4. Identify most likely haplotypes.

The CYP21A2 Caller requires WGS data aligned to a human reference genome with at least 30x coverage.

Total CYP21A2 and CYP21A1P Copy Number

The first step of CYP21A2 calling is to determine the combined copy number of CYP21A2 and CYP21A1P. Reads aligned to regions in either CYP21A2 or CYP21A1P are counted. The counts in each region are corrected for GC-bias, and then normalized to a diploid baseline. The GC-bias correction and normalization factors are determined from read counts in 3000 preselected 2kb regions across the genome. These 3000 normalization regions were randomly selected from the portion of the reference genome having stable coverage across population samples. The combined CYP21A2 and CYP21A1P copy number is then calculated from the average sequencing depth across the CYP21A2 and CYP21A1P regions.

Small Variant Calling

There are 33 common small variants detected by the CYP21A2 caller from the read alignments. Three of these variants are in unique (nonhomologous) regions of CYP21A2 with high mapping quality. Only reads mapping to CYP21A2 are used for calling variants in nonhomologous regions. The other 30 variants occur in homologous regions of CYP21A2/CYP21A1P where reads mapping to either are used for variant calling.

For each variant, reads containing either the variant allele or the nonvariant alleles are counted. A binomial model that incorporates the sequencing errors is then used to determine the most likely variant copy number (0 for nonvariant).

Recombinant Variant Calling

To analyze the homologous region even further, DRAGEN phases reads covering differentiating sites and known variant sites. Whenever a detected haplotype has a CYP21A2->CYP21A1P or CYP21A1P->CYP21A2 transition that is consistent with one of the known recombinant-like variants, the transition is considered as a candidate breakpoint for calling those variants. Reads containing phasing information for the two sites flanking each candidate breakpoint are used for variant calling. When the read data supports the hypothesis that the sample contains at most one copy of the wildtype gene, the variants are considered for calling. All candidates are sorted by likelihood and the number of variant sites. If no wildtype haplotype was detected, DRAGEN calls either one homozygous set of recombinant variants, or two different ones as compound het. If one wildtype haplotype was found, DRAGEN calls a single recombinant variant. Otherwise two wildtype genotypes are output.

CYP21A2 Output File

The output generated by DRAGEN CYP21A2 Caller is part of the targeted callers output <prefix>.targeted.json (that also aggregates calls from other targeted callers).

Fields in JSON	Explanation	Type and Possible Values
totalCopyNumber	Total copy number of CYP21A2+CYP21A1P genes	non-negative integer
deletionBreakpointInGene	Null if total_CN > 3 True if CN <= 3 and a deletion recombinant variant (more than one non-target site) is detected in CYP21A2 False if CN <=3 and no deletion recombinant variant is detected in CYP21A2	True, False, null
recombinantHaplotypes	List of known variants from recombination that were detected in CYP21A2.	Array of recombinant variants with two entries. Each entry contain the IDs of recombinant variants separated by a '+'.
variants	List of single site variants, non-CYP21A1P like variants (not a result of homology). An empty list if no variants are detected.	Array of non-CYP21A1P like variants.

Output File Example

The output file contains information on the sample and different targeted callers. Consider the following example contents.

```
{
  "dragenVersion": "4.2.0-286-g4c94170a",
  "sample": "NA14734",
  "cyp21a2": {
    "totalCopyNumber": 2,
    "deletionBreakpointInGene": true,
    "recombinantHaplotypes": [
      ...
    ]
  }
}
```

```

"NM_000500.9:c.92C>T+NM_000500.9:c.293-13C>G+NM_000500.9:c.332_339del+NM_
000500.9:c.518T>A+NM_000500.9:c.710T>A+NM_000500.9:c.713T>A+NM_
000500.9:c.719T>A+NM_000500.9:c.923dupT",
"NM_000500.9:c.92C>T+NM_000500.9:c.293-13C>G+NM_000500.9:c.332_339del+NM_
000500.9:c.518T>A+NM_000500.9:c.710T>A+NM_000500.9:c.713T>A+NM_
000500.9:c.719T>A+NM_000500.9:c.923dupT+NM_000500.9:c.955C>T"
],
"variants": []
}
}

```

Command-Line Examples

To enable the CYP21A2 Caller, use `--enable-cyp21a2=true`. The CYP21A2 Caller is disabled by default. The CYP21A2 Caller can run directly from FASTQ input with the mapper or from prealigned BAM/CRAM input. You can also enable the CYP21A2 Caller in parallel with any other germline variant callers as part of a WGS germline analysis workflow. For more information on other variant callers, see DNA Pipeline for DRAGEN.

FASTQ Input

The following command-line example uses FASTQ input:

```

dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/8 \
--fastq-file1 /staging/test/data/NA12878_R1.fastq \
--fastq-file2 /staging/test/data/NA12878_R2.fastq \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--RGID DRAGEN_RGID \
--RGSM NA12878 \
--enable-map-align=true \
--enable-cyp21a2=true

```

Prealigned BAM Input

The following command-line example uses BAM input that has already been aligned:

```

dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/8 \
--fastq-file1 /staging/test/data/NA12878_R1.fastq \

```

```
--fastq-file2 /staging/test/data/NA12878_R2.fastq \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--RGID DRAGEN_RGID \
--RGSM NA12878 \
--enable-map-align=true \
--enable-cyp21a2=true
Prealigned BAM Input
```

GBA Caller

The GBA Caller is capable of detecting both recombinant-like and non-recombinant-like variants in the GBA gene from whole-genome sequencing (WGS) data. Disruption of all copies of the GBA gene in an individual causes the autosomal recessive disorder Gaucher disease, and carriers are at increased risk of Parkinson's disease and Lewy body dementia. Due to high sequence similarity with its pseudogene paralog GBAP1, calling recombinant-like variants in GBA requires a specialized caller.

To enable the GBA Caller, use `--enable-gba=true` as part of a germline-only WGS analysis workflow. The GBA Caller is disabled by default and requires WGS data aligned to a human reference genome with at least 30x coverage.

The GBA Caller performs the following steps:

1. Determine the total combined GBA and GBAP1 copy number
2. Detect non-recombinant-like variants from a set of 111 known variants
3. Assemble phased haplotypes in the exon 9-11 region where recombinant variants occur
4. Detect any GBAP1 -> GBA breakpoints that are consistent with one of the 7 known recombinant-like variants

Total Combined GBA and GBAP1 Copy Number

A 10 kb region of unique sequence in between GBA and GBAP1 is used to compute the copy number change due to reciprocal recombination events. Reads that align to this 10 kb region are counted and the count is normalized to a diploid baseline derived from 3000 preselected 2 kb regions across the genome. The 3000 normalization regions are randomly selected from the portion of the reference genome that has stable coverage across population samples. The total combined GBA and GBAP1 copy number is then calculated as two more than the copy number of this 10 kb region.

Non-recombinant-like Variant Calling

Of the 111 known non-recombinant-like variants, 10 are in unique (non-homologous) regions of GBA with high mapping quality. Only reads mapping to GBA is used for calling variants in nonhomologous regions. The other 101 variants occur in homologous regions of GBA/GBAP1 where reads mapping to

either GBA or GBAP1 are used for variant calling.

For each variant, reads containing either the variant allele or the nonvariant alleles are counted. A binomial model that incorporates the sequencing errors is then used to determine the most likely variant copy number (0 for nonvariant). If the posterior probability that the variant copy number is not zero is above 0.9, then the variant is reported in the output CSV file.

Haplotype Phasing in the Exon 9-11 Region

A collection of 10 differentiating sites in the exon 9-11 region of GBA are used to detect the GBA and GBAP1 haplotypes present in the sample. An iterative phasing algorithm is used to build up haplotypes that are supported by the read data. The phasing algorithm starts with seed sites which are then iteratively extended to neighboring sites. At each iteration, reads that can be unambiguously assigned to one of the detected partial haplotypes are used to extend the next neighboring site for each partial haplotype. Iteration continues until all sites have been extended. Some haplotypes may have sites that are unresolved (ie ambiguous), but these haplotypes can still participate in GBA -> GBAP1 breakpoint detection.

Recombinant-like Variant Calling

If any of the 10 differentiating sites in exon 9-11 indicate that there are no wildtype GBA allele copies, then the sample is called a homozygous variant and the recombinant-like variant that best matches the depth calls at the 10 sites is reported.

When the sample is not a homozygous variant, the phased haplotypes are used to detect heterozygous variants. The detected haplotypes are compared against a set of 7 known recombinant-like variants: A495P, L483P, D448H, c.1263del, RecNcil, RecTL, c.1263del+RecTL). Whenever a detected haplotype has a GBA->GBAP1 or GBAP1->GBA transition that is consistent with one of these 7 known recombinant-like variants, the transition is considered as a candidate breakpoint for calling that recombinant-like variant. Reads containing phasing information for the two sites flanking each candidate breakpoint are used for variant calling. When the read data supports the hypothesis that only a single copy of the wildtype GBA gene is present, then the variant is called. A multinomial allele model for the two sites is used to determine the posterior probability that there is one copy of the wildtype GBA gene and a probability threshold is used in deciding to make the variant call. The probability threshold is 0.45 for gene-conversion variants (A495P, L483P, D448H, or c.1263del) when the total GBA and GBAP1 copy number is at most 4, and 0.7 in all other cases.

When the phased haplotypes indicate there are no copies of the wildtype GBA haplotype, then the sample is called a compound heterozygous variant and the two most likely variants are reported.

GBA Caller Output File

The GBA Caller prints out its calls in the targeted callers output file, <prefix>.targeted.json (that also aggregates calls from other targeted callers). An example of this file with the GBA caller set is as follows:

```
{
  "dragenVersion": "4.2.0-758-g4da735e4",
  "sample": "BF-1016",
  "gba": {
    "totalCopyNumber": 4,
    "deletionBreakpointInGene": null,
    "recombinantHaplotypes": [
      "L483P",
      ""
    ],
    "variants": []
  }
}
```

For GBA caller, the fields are defined as follows.

Fields in JSON	Explanation	Type and Possible Values
dragenVersion	Version of DRAGEN	string
sample	sample id	string
gba	a json array containing the gba call for this sample	json-array
totalCopyNumber	Total number of copies of GBA, GBAP1 and hybrids	Nonnegative integer
deletionBreakpointInGene	Indicates if the sample has one of the recombinant-like deletion variants (RecNcil, RecTL, or c.1263del+RecTL). null is reported if totalCopyNumber > 3.	true false null

Fields in JSON	Explanation	Type and Possible Values
recombinantHaplotypes	List of detected recombinant-like variants in GBA: A495P, L483P, D448H, 1263del, RecNcil, RecTL, or c.1263del+RecTL.	string (list of recombinant-like variants)
variants	List of single site non-recombinant-like variants in GBA. An empty list if no variants are detected.	string (list of target variants found)

The GBA Caller generates a `<output-file-prefix>.gba.tsv` file in the output directory when the option `--targeted-enable-legacy-output=true` is set. The output file contains a header line followed by a sample call line that contains the following tab-delimited fields.

Field Header	Description	Value(s)
Sample	Sample name	String
is_biallelic	Presence of a recombinant-like variant on each chromosome (homozygous variant or compound heterozygous)	<ul style="list-style-type: none"> • True • False
is_carrier	Presence of a recombinant-like variant on only one chromosome	<ul style="list-style-type: none"> • True • False
total_CN	Total number of copies of GBA, GBAP1 and hybrids	<ul style="list-style-type: none"> • Non negative integer
deletion_breakpoint_in_GBA_gene	Indicates if the sample has one of the recombinant-like deletion variants (RecNcil, RecTL, or c.1263del+RecTL). None is reported if total_CN > 3.	<ul style="list-style-type: none"> • True • False • None
recombinant_variants	List of detected recombinant-like variants in GBA: A495P, L483P, D448H, 1263del, RecNcil, RecTL, or c.1263del+RecTL. If is_biallelic is True then two variants will be identified. If the sample is a carrier, one variant will be identified. Otherwise the field will contain an empty list.	Slash separated list of recombinant-like variants
other_variants	List of single site non-recombinant-like variants in GBA. An empty list if no variants are detected.	Comma separated list of variant identifiers

For a list of the 111 supported non-recombinant-like variants, refer to the *GBA_target_variant_*.tsv* files located in the config directory of the DRAGEN install location.

The following is an example <output-file-prefix>.gba.tsv output file.

#Sample	is_biallelic	is_carrier	total_CN	deletion_breakpoint_in_GBA_gene	recombinant_variants	other_variants
LB-00009	False	True	4	None	RecNcil	p.Val499Leu

Command-line Examples

To enable the GBA Caller, use `--enable-gba=true`. The GBA Caller is disabled by default. The GBA Caller can run from FASTQ input with the mapper enabled or from prealigned BAM/CRAM input. You can also enable the GBA Caller in parallel with any other germline variant callers for a WGS germline analysis workflow. For information on other variant callers, see [DRAGEN DNA Pipeline on page 79](#).

FASTQ Input

The following command-line example uses FASTQ input.

```
dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/8 \
--fastq-file1 /staging/test/data/NA12878_R1.fastq \
--fastq-file2 /staging/test/data/NA12878_R2.fastq \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--RGID DRAGEN_RGID \
--RGSM NA12878 \
--enable-map-align=true \
--enable-gba=true
```

Prealigned BAM Input

The following command-line example uses BAM input that has already been aligned.

```
dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/8 \
--bam-input /staging/test/data/NA12878.bam \
--output-directory /staging/test/output \
```

```
--output-file-prefix NA12878_dragen \
--enable-map-align=false \
--enable-gba=true
```

HBA Caller

The HBA Caller is capable of genotyping the HBA1 and HBA2 genes from whole-genome sequencing (WGS) data. Due to high sequence similarity between the genes, a specialized caller is necessary to resolve the possible genotypes of the pair of genes. We consider regions surrounding the HBA1 and HBA2 sites to resolve the possible HBA1 and HBA2 genotypes.

The HBA Caller performs the following steps:

1. Determines total copy number from read depth of the regions surrounding the HBA1 and HBA2 sites.
2. Determines HBA genotypes based on the copy number of the regions surrounding the HBA1 and HBA2 sites.
3. Calls small variants in the HBA1 and HBA2 regions based on the region copy number derived from the genotype along with allele counts from read information.

The HBA Caller requires WGS data aligned to a human reference genome with at least 30x coverage. Reference genome builds must be based on hg19, GRCh37, or hg38.

For a comprehensive evaluation of the HBA caller, we refer to the HBA targeted caller blog post.

Total Copy Number of the regions surrounding the HBA1 and HBA2 sites

The first step of HBA calling is to determine the copy number of the regions surrounding the HBA1 and HBA2 sites. Reads aligned to the regions are counted. The counts in each region are corrected for GC-bias, and then normalized to a diploid baseline. The GC-bias correction and normalization factors are determined from read counts in 3000 preselected 2 kb regions across the genome. These 3000 normalization regions were randomly selected from the portion of the reference genome having stable coverage across population samples. Finally, a Gaussian Mixture Model (GMM) is used to obtain the integer region copy number from the region normalized counts.

Genotyping

The genotyping step attempts to identify the two likely haplotypes described in the following table, where "a" stands for a functional copy of either HBA1 or HBA2, "-" stands for a non-functional/missing copy of either HBA1 or HBA2, while "3.7" and "4.2" describe the recombinant event that likely caused the deletion/duplication of the functional HBA copy. The second column of the following table reports the interpretation of the genotype.

Genotype	Interpretation
aaa3.7/aa	alpha-globin triplication
aaa4.2/aa	alpha-globin triplication
aa/aa	Normal
-a3.7/aa	Silent Carrier
-a4.2/aa	Silent Carrier
--/aaa3.7	Carrier
--/aaa4.2	Carrier
-a3.7/-a3.7	Carrier
-a4.2/-a4.2	Carrier
-a3.7/-a4.2	Carrier
--/aa	Carrier
--/-a3.7	HbH
--/-a4.2	HbH
--/--	Hb Bart's

If none of the previous genotype is identified, then no call is made and the caller reports a `None` genotype.

Small Variant Calling

18 small variants are detected from the read alignments. These variants occur in homologous regions of HBA1 and HBA2 where reads mapping to either HBA1 or HBA2 are used for variant calling.

For each variant, reads containing either the variant allele or the non-variant allele are counted and a binomial model is used to determine the likelihood for each possible variant allele copy number up to the maximum possible as determined from the HBA1/HBA2 genotyping.

HBA Output File

The HBA Caller generates a <output-file-prefix>.targeted.json file in the output directory. The output file is a JSON formatted file containing the fields below.

Fields in JSON	Explanation	Type and Possible Values
sample	The sample name.	string
dragenVersion	The version of DRAGEN.	string
hba	The HBA targeted caller specific fields.	dictionary

The hba fields are defined as below.

Fields in JSON	Explanation	Type and Possible Values
genotype	The HBA genotype.	string
genotypeFilter	The HBA genotype filter.	string, [PASS, HBALowGQ, HBALowPValue, No_call]
genotypeQual	The HBA Phred genotype quality.	double
minPValue	The minimum copy number p-value of regions used to determine copy number genotype of the HBA locus.	double
variants	List of known variants that were detected in HBA1/HBA2.	list of variants

For the variants the fields are defined as below.

Fields in JSON	Explanation	Type and Possible Values
hgvs	HGVS identifier of the variant	string
qual	Phred QUAL score of the variant	double
altCopyNumber	Copy number of the ALT variant	double
altCopyNumberQuality	Phred QUAL copy number of the ALT variant	double

The following are example output files:

```
{  
  "dragenVersion": "4.2.0",  
  "sample": "HG00699",  
  "hba": {  
    "genotype": "--/aa",  
    "genotypeFilter": "PASS",  
    "genotypeQual": 96.70607775855007,  
    "minPValue": 0.0009718284337509549,  
    "variants": [  
      {  
        "hgvs": "NM_000517.6:c.60delG",  
        "qual": 0.0,  
        "altCopyNumber": 0,  
        "altCopyNumberQuality": 150.0  
      },  
      ...  
      {  
        "hgvs": "NM_000517.6:c.*94A>G",  
        "qual": 0.0,  
        "altCopyNumber": 0,  
        "altCopyNumberQuality": 150.0  
      }  
    ]  
  }  
}
```

```
}
```

```
{
  "dragenVersion": "4.2.0",
  "sample": "HG04038",
  "hba": {
    "genotype": "aa/aa",
    "genotypeFilter": "PASS",
    "genotypeQual": 97.70012179185903,
    "minPValue": 0.050571535072591677,
    "variants": [
      {
        "hgvs": "NM_000517.6:c.60delG",
        "qual": 0.0,
        "altCopyNumber": 0,
        "altCopyNumberQuality": 150.0
      },
      ...
      {
        "hgvs": "NM_000517.6:c.95+1G>A",
        "qual": 150.0,
        "altCopyNumber": 1,
        "altCopyNumberQuality": 20.92247444858537
      }
    ]
  }
}
```

```
        },
        ...
    {
        "hgvs": "NM_000517.6:c.*94A>G",
        "qual": 0.0,
        "altCopyNumber": 0,
        "altCopyNumberQuality": 150.0
    }
]
}
```

```
{
    "dragenVersion": "4.2.0",
    "sample": "HG00635",
    "hba": {
        "genotype": "None",
        "genotypeFilter": "No_call",
        "genotypeQual": 23.423375519510774,
        "minPValue": 0.11387347538621106,
        "variants": []
    }
}
```

The HBA Caller also generates a <output-file-prefix>.targeted.vcf[.gz] file in the output directory. The output file is a VCFv4.2 formatted file possibly compressed.

The output file contains the VCF representation of the target variants and the structural variants inferred from the identified genotype. The target variant are reported with EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION in the INFO field and also filtered with the TargetedRepeatConflict filter. The target variants are reported on both HBA1 and HBA2 regions and connected by the same EVENT in the INFO field. The ploidy of the variant is reported in concordance with the identified genotype.

The HBALowQUAL filter can be applied to the structural variant records if the genotype call has a QUAL < 3.00. The filters are specified by the following header line.

```
##FILTER=<ID=HBALowQUAL,Description="Set if HBA genotype call has QUAL < 3.00">
```

The following are example output files:

```
##fileformat=VCFv4.2

...
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT HG00699

chr16 165397 . A <DEL> 150.00 PASS END=184700;IMPRECISE;SVLEN=19303;CI
POS=0,0;SVCLAIM=D;EVENT=HBA:--/aa;CN=0 GT:GQ:PS

1|.:97:165397

chr16 165397 . A <CNV> 150.00 PASS END=184700;IMPRECISE;SVLEN=19303;CI
POS=0,0;SVCLAIM=D;EVENT=HBA:--/aa;CN=1 GT:GQ:PS

.|0:97:165397

chr16 172971 . CG C 0.00 TargetedLowQual;TargetedRepeatConflict EVENT
=NM_000517.6:c.60delG;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION GT:GQ 0/0:150

chr16 172981 . C T 0.00 TargetedLowQual;TargetedRepeatConflict EVENT
=NM_000517.6:c.69C>T;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION GT:GQ 0/0:150

...
chr16 176775 . CG C 0.00 TargetedLowQual;TargetedRepeatConflict EVENT
=NM_000517.6:c.60delG;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION GT:GQ 0/0:150

chr16 176785 . C T 0.00 TargetedLowQual;TargetedRepeatConflict EVENT
=NM_000517.6:c.69C>T;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION GT:GQ 0/0:150

...
```

```
chr16 177504 . A G 0.00 TargetedLowQual;TargetedRepeatConflict EVENT
=NM_000517.6:c.*92A>G;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION GT:GQ 0/0:150
```

```
chr16 177506 . A G 0.00 TargetedLowQual;TargetedRepeatConflict EVENT
=NM_000517.6:c.*94A>G;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION GT:GQ 0/0:150
```

```
##fileformat=VCFv4.2
```

```
...
```

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	HG04038
chr16	165397	.	A	<CNV>	0.00	HBALowQUAL		END=184700;IMPRECISE;SVLEN=	
							GT:GQ		

```
0/0:97
```

```
chr16 172971 . CG C 0.00 TargetedLowQual;TargetedRepeatConflict EVENT
=NM_000517.6:c.60delG;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION GT:GQ 0/0/0/0:
```

```
150
```

```
...
```

chr16	173008	.	G	A	150.00	TargetedRepeatConflict	EVENT=NM_000517.6:c.95+1G>A;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION	GT:GQ	
-------	--------	---	---	---	--------	------------------------	--	-------	--

```
0/0/0/1:21
```

```
...
```

chr16	176812	.	G	A	150.00	TargetedRepeatConflict	EVENT=NM_000517.6:c.95+1G>A;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION	GT:GQ	
-------	--------	---	---	---	--------	------------------------	--	-------	--

```
0/0/0/1:21
```

```
...
```

chr16	177506	.	A	G	0.00	TargetedLowQual;TargetedRepeatConflict	EVENT=NM_000517.6:c.*94A>G;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION	GT:GQ	0/0/0/0:
-------	--------	---	---	---	------	--	---	-------	----------

```
150
```

```
##fileformat=VCFv4.2
...
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT HG00635
```

Command-line Examples

To enable the HBA Caller, use `--enable-hba=true`. The HBA Caller is disabled by default. The HBA Caller can run directly from FASTQ input with the mapper or from prealigned BAM/CRAM input. You can also enable the HBA Caller in parallel with any other germline variant callers as part of a WGS germline analysis workflow. For more information on other variant callers, see DNA Pipeline for DRAGEN.

The targeted variants in the `<output-file-prefix>.targeted.vcf[.gz]` file can be included in the `<output-file-prefix>.hard-filtered.vcf[.gz]` by including the `hba` entry in the `--targeted-merge-vc` list, i.e. `--targeted-merge-vc hba`. The targeted variants included in the `<output-file-prefix>.hard-filtered.vcf[.gz]` are marked with a `TARGETED` flag in the `INFO` field.

The output file `<output-file-prefix>.targeted.vcf[.gz]` is compressed by default. This option can be disabled using `--enable-vcf-compression=false`.

FASTQ Input

The following command-line example uses FASTQ input:

```
dragen \
-r /staging/human/reference/hg38_1_alt_masked_v3/DRAGEN/9 \
--fastq-file1 /staging/test/data/NA12878_R1.fastq \
--fastq-file2 /staging/test/data/NA12878_R2.fastq \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--RGID DRAGEN_RGID \
--RGSM NA12878 \
--enable-map-align=true \
--enable-hba=true
```

Prealigned BAM Input with VCF merge enabled

The following command-line example uses BAM input that has already been aligned:

```
dragen \
-r /staging/human/reference/hg38_1_alt_masked_v3/DRAGEN/9 \
--bam-input /staging/test/data/NA12878.bam \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--enable-map-align=false \
--enable-hba=true \
--targeted-merge-vc hba
```

LPA Caller

The LPA Caller is capable of identifying the LPA Kringle-IV-2 (KIV-2) VNTR unit copy number from whole-genome sequencing (WGS) data. Due to high sequence similarity between the genes, a specialized caller is necessary to resolve the VNTR unit copy number.

The LPA Caller performs the following steps:

1. Determines total LPA KIV-2 VNTR unit copy number.
2. Determines the heterozygous LPA KIV-2 VNTR unit copy number if heterozygous KIV-2 markers are present.
3. Calls small variants in the LPA KIV-2 VNTR region based on the unit copy number along with allele counts from read information.

The LPA Caller requires WGS data aligned to a human reference genome with at least 30x coverage. Reference genome builds must be based on hg19, GRCh37, or hg38.

Total LPA KIV-2 VNTR Unit Copy Number

The first step of LPA calling is to determine the unit copy number of LPA KIV-2. Reads aligned to the LPA KIV-2 are counted. The counts in each region are corrected for GC-bias, and then normalized to a diploid baseline. The GC-bias correction and normalization factors are determined from read counts in 3000 preselected 2 kb regions across the genome. These 3000 normalization regions were randomly selected from the portion of the reference genome having stable coverage across population samples.

Heterozygous LPA KIV-2 VNTR Unit Copy Number

The second step of LPA calling is to determine the heterozygous unit copy number of LPA KIV-2. Heterozygous unit copy number is determined using two specific linked SNV sites that have been identified as a combined marker allele that is always present in every copy of the repeat unit concordantly. That is, if any copy of the repeat unit in an LPA haplotype contains the ALT alleles at those two SNV sites, then every copy of the repeat unit in that LPA haplotype contains the ALT alleles at those two sites. The relative read coverage for the ALT and REF cases at these sites can therefore be used to determine the proportions of overall copy numbers across the KIV repeat array that belong to each haplotype.

Small Variant Calling

2 small variants are detected from the read alignments. These variants occur in the LPA KIV-2 VNTR region where reads mapping to either of the 6 units in the reference are used for variant calling.

For each variant, reads containing either the variant allele or the non-variant allele are counted and a binomial model is used to determine the likelihood for each possible variant allele copy number up to the maximum possible as determined from the LPA KIV-2 VNTR unit copy number.

LPA Output File

The LPA Caller generates a `<output-file-prefix>.targeted.json` file in the output directory. The output file is a JSON formatted file containing the fields below.

Fields in JSON	Explanation	Type and Possible Values
sample	The sample name.	string
dragenVersion	The version of DRAGEN.	string
lpa	The LPA targeted caller specific fields.	dictionary

The `lpa` fields are defined as below.

Fields in JSON	Explanation	Type and Possible Values
kiv2CopyNumber	Total KIV-2 unit copy number	float
refMarkerAlleleCopyNumber	Null if Homozygous REF/ALT markers call	float, null

Fields in JSON	Explanation	Type and Possible Values
	Float if Heterozygous markers call and stores the KIV-2 unit copy number of the allele having REF markers	
altMarkerAlleleCopyNumber	Null if Homozygous REF/ALT markers call	float, null
	Float if Heterozygous markers call and stores the KIV-2 unit copy number of the allele having ALT markers	
type	"Heterozygous markers call" if we observe both REF and ALT markers "Homozygous REF markers call" if we observe only REF markers "Homozygous ALT markers call" if we observe only ALT markers	string, "Heterozygous markers call", "Homozygous REF markers call", "Homozygous ALT markers call"
variants	List of known variants that were detected in the KIV-2 region.	list of variants

For the `variants` the fields are defined as below.

Fields in JSON	Explanation	Type and Possible Values
hgvs	HGVS identifier of the variant	string

Fields in JSON	Explanation	Type and Possible Values
qual	Phred QUAL score of the variant	double
altCopyNumber	Copy number of the ALT variant	double
altCopyNumberQuality	Phred QUAL copy number of the ALT variant	double

The LPA Caller also generates a <output-file-prefix>.targeted.vcf[.gz] file in the output directory. The output file is a VCFv4.2 formatted file possibly compressed.

JSON Output File Examples

The following are example output files:

```
{
  "dragenVersion": "4.2.0",
  "sample": "HG00096",
  "lpa": {
    "kiv2CopyNumber": 39.236019179048256,
    "refMarkerAlleleCopyNumber": 17.84675290975861,
    "altMarkerAlleleCopyNumber": 21.389266269289646,
    "type": "Heterozygous markers call",
    "variants": [
      {
        "hgvs": "LPA:4925G>A",
        "qual": 0.0,
        "altCopyNumber": 0,
        "altCopyNumberQuality": 150.0
      },
      {
        "hgvs": "LPA:4925G>A",
        "qual": 0.0,
        "altCopyNumber": 0,
        "altCopyNumberQuality": 150.0
      }
    ]
  }
}
```

```
        "hgvs": "LPA:4733G>A",
        "qual": 135.16839207370617,
        "altCopyNumber": 1,
        "altCopyNumberQuality": 18.641166763477106
    }
]
}
```

```
{
  "dragenVersion": "4.2.0-390-g36e63007",
  "sample": "HG00097",
  "lpa": {
    "kiv2CopyNumber": 28.709054093341653,
    "refMarkerAlleleCopyNumber": null,
    "altMarkerAlleleCopyNumber": null,
    "type": "Homozygous REF markers call",
    "variants": [
      {
        "hgvs": "LPA:4925G>A",
        "qual": 0.0,
        "altCopyNumber": 0,
        "altCopyNumberQuality": 150.0
      }
    ]
  }
}
```

```
    },
    {
        "hgvs": "LPA:4733G>A",
        "qual": 132.8341692747592,
        "altCopyNumber": 1,
        "altCopyNumberQuality": 15.776967125542765
    }
}
```

VCF Output File Examples

The following are example output files:

Command-line Examples

To enable the LPA Caller, use `--enable-lpa=true`. The LPA Caller is disabled by default. The LPA Caller can run directly from FASTQ input with the mapper or from prealigned BAM/CRAM input. You can also enable the LPA Caller in parallel with any other germline variant callers as part of a WGS germline analysis workflow. For more information on other variant callers, see DNA Pipeline for DRAGEN.

The variants in the <output-file-prefix>.targeted.vcf[.gz] file can be included in the <output-file-prefix>.hard-filtered.vcf[.gz] by including the lpa entry in the --targeted-merge-vc list, i.e. --targeted-merge-vc lpa. The targeted variants included in the <output-file-prefix>.hard-filtered.vcf[.gz] are marked with a TARGETED flag in the INFO field.

The output file `<output-file-prefix>.targeted.vcf[.gz]` is compressed by default. This option can be disabled using `--enable-vcf-compression=false`.

FASTQ Input

The following command-line example uses FASTQ input:

```
dragen \
-r /staging/human/reference/hg38_1_alt_masked_v3/DRAGEN/9 \
--fastq-file1 /staging/test/data/NA12878_R1.fastq \
--fastq-file2 /staging/test/data/NA12878_R2.fastq \
--output-directory /staging/test/output \
```

```
--output-file-prefix NA12878_dragen \
--RGID DRAGEN_RGID \
--RGSM NA12878 \
--enable-map-align=true \
--enable-lpa=true
```

Prealigned BAM Input with VCF merge enabled

The following command-line example uses BAM input that has already been aligned:

```
dragem \
-r /staging/human/reference/hg38_1_alt_masked_v3/DRAGEN/9 \
--bam-input /staging/test/data/NA12878.bam \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragem \
--enable-map-align=false \
--enable-lpa=true \
--targeted-merge-vc lpa
```

Rh Caller

The Rh Caller is capable of identifying a common gene conversion between RHD and RHCE genes from whole-genome sequencing (WGS) data, that is referred to as RHCE Exon2 gene conversion. Due to high sequence similarity between the genes, a specialized caller is necessary to resolve the gene conversion between the pair of genes. We consider 798 loci, called differentiating sites, that represents differences between the RHD and RHCE genes, that are well preserved in the population.

The Rh Caller performs the following steps:

1. Determines total copy number from read depth of the RHD and RHCE regions.
2. Detect RHD -> RHCE breakpoints that are consistent with the RHCE Exon2 gene conversion.

The Rh Caller requires WGS data aligned to a human reference genome with at least 30x coverage. Reference genome builds must be based on hg19, GRCh37, or hg38.

The Rh Caller is run by default when the small variant caller is enabled, the sample is a non-tumor sample, and the sample is detected as WGS by the Ploidy Estimator.

Total Combined RHD and RHCE Copy Number

The first step of Rh calling is to determine the copy number of RHD and RHCE regions. Reads aligned to the RHD and RHCE regions are counted according to their support of the differentiating sites. The counts in each region are corrected for GC-bias, and then normalized to a diploid baseline. The GC-bias correction and normalization factors are determined from read counts in 3000 preselected 2 kb regions across the genome. These 3000 normalization regions were randomly selected from the portion of the reference genome having stable coverage across population samples.

Haplotype Phasing in the Exon 2 Region

A collection of 4 differentiating sites in the exon 2 region of RHD and RHCE are used to detect the presence of the RHCE Exon2 gene conversion in the sample. An iterative phasing algorithm is used to build up haplotypes that are supported by the read data. The phasing algorithm starts with candidate haplotypes formed from all possible bases at the first differentiating site. The haplotypes are then extended at the next differentiating site by considering all reads that can be uniquely assigned to a single candidate haplotype. If these reads support only a single base at the next differentiating site for a given candidate haplotype, then the haplotype is extended with that base. When a candidate haplotype can be extended by both bases at the next differentiating site then both possible extended haplotypes are included in the set of candidate haplotypes, growing the set by 1. Subsequent extension steps are performed at neighboring differentiating sites until all sites have been processed. Some haplotypes may have sites that are unresolved (i.e. ambiguous), but these haplotypes can still participate in RHD -> RHCE breakpoint detection.

Recombinant-like Variant Calling

When the phased haplotypes support the RHCE Exon2 gene conversion. We visit all the differentiating sites ad report them as variants in the output VCF file with ploidy identified using the copy number estimated from the read depth of the differentiating site.

Rh Output File

The Rh Caller generates a <output-file-prefix>.targeted.json file in the output directory. The output file is a JSON formatted file containing the fields below.

Fields in JSON	Explanation	Type and Possible Values
sample	The sample name.	string
dragenVersion	The version of DRAGEN.	string
rh	The RH targeted caller specific fields.	dictionary

The `rh` fields are defined as below.

Fields in JSON	Explanation	Type and Possible Values
totalCopyNumber	Total RHD/RHCE copy number	integer
rhdCopyNumber	RHD gene copy number	integer
rhceCopyNumber	RHCE copy number	integer
variants	List of known variants from recombination that were detected in RHD/RHCE.	list of variants

For the `variants` the fields are defined as below.

Fields in JSON	Explanation	Type and Possible Values
hgvs	HGVS identifier of the variant	string, "NC_000001.11g.25405596_25409676con25283766_25287797"
qual	Phred QUAL score of the variant	double
altCopyNumber	Copy number of the ALT variant	double
altCopyNumberQuality	Phred QUAL copy number of the ALT variant	double

The following are example output files:

```
{
  "dragenVersion": "4.2.0",
  "sample": "HG002",
  "rh": {
    "totalCopyNumber": 4,
    "rhdCopyNumber": 2,
    "rhceCopyNumber": 2,
    "variants": [
      ...
    ]
  }
}
```

```
{  
    "hgvs": "NC_00001.11g.25405596_25409676con25283766_25287797",  
    "qual": 63.42360372675952,  
    "altCopyNumber": 2,  
    "altCopyNumberQuality": 74.27989942583051  
}  
]  
}  
}
```

```
{  
    "dragenVersion": "4.2.0",  
    "sample": "HG00110",  
    "rh": {  
        "totalCopyNumber": 4,  
        "rhdCopyNumber": 2,  
        "rhceCopyNumber": 2,  
        "variants": [  
            {  
                "hgvs": "NC_00001.11g.25405596_25409676con25283766_25287797",  
                "qual": 0.0,  
                "altCopyNumber": 0,  
                "altCopyNumberQuality": 0.026368888668874203  
            }  
        ]  
    }  
}
```

```

    }
]
}
}
```

The Rh Caller also generates a <output-file-prefix>.targeted.vcf[.gz] file in the output directory. The output file is a VCFv4.2 formatted file, possibly compressed.

The following are example output files:

```

##fileformat=VCFv4.2
...
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT HG002
chr1 25405596 . G A 128.19 PASS EVENT=NC_000001.11g.25405596_25409676con25283766_
25287797;EVENTTYPE=GENE_CONVERSION GT:GQ 1/1:128

chr1 25405674 . A T 134.41 PASS EVENT=NC_000001.11g.25405596_25409676con25283766_
25287797;EVENTTYPE=GENE_CONVERSION GT:GQ 1/1:134

...
chr1 25409655 . G C 133.16 PASS EVENT=NC_000001.11g.25405596_25409676con25283766_
25287797;EVENTTYPE=GENE_CONVERSION GT:GQ 1/1:133

chr1 25409676 . G A 104.46 PASS EVENT=NC_000001.11g.25405596_25409676con25283766_
25287797;EVENTTYPE=GENE_CONVERSION GT:GQ 1/1:104
```

```

##fileformat=VCFv4.2
...
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT HG00110
chr1 25405596 . G A 44.96 PASS EVENT=NC_000001.11g.25405596_25409676con25283766_
25287797;EVENTTYPE=GENE_CONVERSION GT:GQ 0/0:45
```

```

chr1 25405674 . A T 35.71 PASS EVENT=NC_000001.11g.25405596_25409676con25283766_
25287797;EVENTTYPE=GENE_CONVERSION GT:GQ 0/0:36

...
chr1 25409655 . G C 32.72 PASS EVENT=NC_000001.11g.25405596_25409676con25283766_
25287797;EVENTTYPE=GENE_CONVERSION GT:GQ 0/0:33

chr1 25409676 . G A 27.11 PASS EVENT=NC_000001.11g.25405596_25409676con25283766_
25287797;EVENTTYPE=GENE_CONVERSION GT:GQ 0/0:27

```

Command-line Examples

To enable the Rh Caller, use `--enable-rh=true`. The Rh Caller is disabled by default. The Rh Caller can run directly from FASTQ input with the mapper or from prealigned BAM/CRAM input. You can also enable the Rh Caller in parallel with any other germline variant callers as part of a WGS germline analysis workflow. For more information on other variant callers, see DNA Pipeline for DRAGEN.

The variants in the `<output-file-prefix>.targeted.vcf[.gz]` file are included in the `<output-file-prefix>.hard-filtered.vcf[.gz]` by default. This option can be disabled by removing the `rh` entry from the `--targeted-merge-vc` or using `--targeted-merge-vc none`. The target variants are reported on the RHCE region and connected by the same `EVENT` in the `INFO` field.

The output file `<output-file-prefix>.targeted.vcf[.gz]` is compressed by default. This option can be disabled using `--enable-vcf-compression=false`.

FASTQ Input

The following command-line example uses FASTQ input:

```

dragen \
    -r /staging/human/reference/hg38_alt_aware/DRAGEN/8 \
    --fastq-file1 /staging/test/data/NA12878_R1.fastq \
    --fastq-file2 /staging/test/data/NA12878_R2.fastq \
    --output-directory /staging/test/output \
    --output-file-prefix NA12878_dragen \
    --RGID DRAGEN_RGID \

```

```
--RGSM NA12878 \
--enable-map-align=true \
--enable-rh=true
```

Prealigned BAM Input with VCF merge disabled

The following command-line example uses BAM input that has already been aligned:

```
dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/8 \
--bam-input /staging/test/data/NA12878.bam \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--enable-map-align=false \
--enable-rh=true \
--targeted-merge-vc none
```

Spinal Muscular Atrophy Calling

Disruption of all copies of the SMN1 gene in an individual causes spinal muscular atrophy (SMA). SMN1 has a high identity paralog, SMN2, which differs only in approximately 10 SNVs and small indels. One of these (hg19 chr5:70247773 C->T) affects splicing and largely disrupts the production of functional SMN protein from SMN2. Standard WGS analysis does not produce complete variant calling results for SMN due to this high-similarity duplication combined with common copy-number variation. However, approximately 95% of SMA cases can be detected by determining the absence of the functional C (SMN1) allele in any copy of SMN.

DRAGEN SMA calling uses sequence-graph realignment to align reads to a single reference representing SMN1 and SMN2. In addition to the standard diploid genotype call, DRAGEN uses a direct statistical test to check for presence of any C allele. If no C allele is detected, the sample is called affected, otherwise unaffected.

When DRAGEN 3.9+ is used for secondary analysis the SMN1 Copy Number, SMN2 Copy Number, and Carrier Status will be reported as additional metrics in the variant details.

SMA calling is only supported for human whole-genome sequencing samples in PCR-free libraries.

Disruption of all copies of the SMN1 gene in an individual causes spinal muscular atrophy (SMA). SMN1 has a high identity paralog, SMN2. SMN2 differs only in approximately 10 SNVs and small indels. For example, hg19 chr5:70247773 C->T affects splicing and largely disrupts the production of functional SMN protein from SMN2. Due to the high-similarity duplication combined with common-copy number variation, standard whole-genome sequencing (WGS) analysis does not produce complete variant calling results for SMN. Since 95% of SMA cases result from the absence of the functional C (SMN1) allele in any copy of SMN¹, a targeted calling solution can be effective in detecting SMA.

DRAGEN offers the following two independent components that can call the SMN1 copy number using WGS data from a germline sample.

- ExpansionHunter
- SMN Caller

¹ WirthB. An update of the mutation spectrum of the survival motor neuron gene (SMN1) in autosomal recessive spinal muscular atrophy (SMA). *Human Mutation*. 2000;15(3):228-237.
doi:10.1002/(sici)1098-1004(200003)15:3<228::aid-humu3>;3.0.co;2-9

SMA Calling With ExpansionHunter

SMA calling is implemented together with repeat expansion detection using sequence-graph realignment to align reads to a single reference that represents SMN1 and SMN2.

In addition to the standard diploid genotype call, SMA Calling with ExpansionHunter uses a direct statistical test to check for presence of any C allele. If a C allele is not detected, the sample is called affected, otherwise unaffected.

SMA calling is only supported for human whole-genome sequencing with PCR-free libraries.

To enable SMA calling along with repeat expansion detection, set the --repeat-genotype-enable option to true. For information on graph-alignment options, see [Repeat Expansion Detection with ExpansionHunter on page 228](#).

To activate SMA calling, the variant specification catalog file must include a description of the targeted SMN1/SMN2 variant. The /opt/edico/repeat-specs/experimental folder contains example files.

The <outputPrefix>.repeat.vcf file includes SMN output along with any targeted repeats. SMN output is represented as a single SNV call at the splice-affecting position in SMN1 with SMA status in the following custom fields.

Table 7 SMA Results in repeat.vcf Output File

Field	Description
VARID	SMN marks the SMN call.
GT	Genotype call at this position using a normal (diploid) genotype model.
DST	SMA status call: + indicates detected - indicates undetected ? indicates undetermined
AD	Total read counts that support the C and T allele.
RPL	Log10 likelihood ratio between the unaffected and affected models. Positive scores indicate the unaffected model is more likely.

SMN Caller

The SMN Caller calls SMN1 and SMN2 copy numbers and detects the presence of a SNP, c.3+80T>G that is associated with SMA silent carrier status. The caller is derived from the method implemented in *Spinal muscular atrophy diagnosis and carrier screening from genome sequencing data*.¹

To enable the SMN Caller, use `--enable-smn=true` as part of a germline-only WGS analysis workflow. Additionally, it can also be set along with other DRAGEN targeted callers by using the option `--enable-targeted=true`. The SMN Caller is disabled by default.

The SMN Caller performs the following steps:

1. Determines total and intact SMN copy numbers
2. Calls SMN1 copy number at eight differentiating sites
3. Determines copy number for one SNP c.*3+80T>G that is associated with silent carrier status

The SMN Caller requires WGS data aligned to a human reference genome with at least 30x coverage.

Total and Intact SMN Copy Number

Two common copy-number variants (CNVs) in SMN1 and SMN2 include whole gene CNV and a partial gene deletion of exons 7 and 8.

Reads that align to either SMN1 or SMN2 are counted. The read counts in exon 1 through exon 6 are used to determine total SMN copy number. The read counts in exon 7 and 8 are used to determine the SMN copies that do not have the exon 7 and 8 deletion (intact SMN copy number). To estimate the SMN copy number for these two regions, read counts are normalized to a diploid baseline derived from 3000 preselected 2 kb regions across the genome. The 3000 normalization regions are randomly selected from the portion of the reference genome that has stable coverage across population samples. The SMN Caller then calculates the number of SMN copies that have the exon 7 and 8 deletion by subtracting the intact SMN copy number from the total SMN copy number.

SMN1 Copy Number at Differentiating Sites

To calculate the SMN1 copy number, the caller uses eight predefined differentiating sites in exons 7 and 8 of SMN1 and SMN2. One of these sites is the splice site variant used for SMA calling with ExpansionHunter (see [SMA Calling With ExpansionHunter on page 283](#)). The caller selects differentiating sites at positions that have sequence differences between SMN1 and SMN2 where calling the SMN1 copy number is most likely to be correct based on sequencing data from the 1000 Genomes Project.

For each differentiating site, the SMN1-specific and SMN2-specific alleles are counted in reads mapping to either SMN1 or the homologous region in SMN2. The caller uses a binomial model to calculate the likelihood of each possible SMN1 copy number from the two gene-specific counts given the intact SMN copy number calculated in the previous step.

Copy number call for silent carrier SNP c.*3+80T>G

The SNP c.*3+80T>G (also referred to as g.27134T>G in literature) is associated with “silent” carriers of SMA when the copy number of SMN1 is 2 i.e., both copies of the SMN1 gene are located on the same haplotype and the other haplotype has a deletion of the gene. This variant is located on intron 7 of SMN1 and its copy number is computed using the same strategy as used for calling the copy number at differentiating sites.

SMN Output File

The SMN Caller prints out its calls in the targeted callers output file, <prefix>.targeted.json (that also aggregates calls from other targeted callers). An example of this file with the SMN caller set is as follows:

```
{
  "dragenVersion": "4.2.0-724-gb600fcef",
  "sample": "NA19374",
  "smn": {
    "smn1CopyNumber": 3,
    "smn2CopyNumber": 1,
    "smn2Delta78CopyNumber": 0,
    "totalCopyNumber": "3.89",
  }
}
```

```

    "fullLengthCopyNumber": "4.10",

    "variants": [
        {
            "hgvs": "NM_000344.4:c.*3+80T>G",
            "qual": null,
            "altCopyNumber": 1,
            "altCopyNumberQuality": 54.63854344932561
        }
    ]
}

}

```

For SMN caller, the fields are defined as follows.

Fields in JSON	Explanation	Type and Possible Values
dragenVersion	Version of DRAGEN	string
sample	sample id	string
smn	a json array containing the SMN call for this sample	json-array
smn1CopyNumber	Copy number of SMN1	non-negative integer
smn2CopyNumber	Copy number of SMN2	non-negative integer
smn2Delta78CopyNumber	Copy number of SMN2Δ7–8 (deletion of exon 7 and 8)	non-negative integer

Fields in JSON	Explanation	Type and Possible Values
totalCopyNumber	Raw normalized depth of total SMN (exons 1 to 6)	non-negative floating point number
fullLengthCopyNumber	Raw normalized depth of intact SMN (exons 7 & 8)	non-negative integer
variants	a json array containing info about specific SMN variants	json-array
hgvs	HGVS id of the variant being reported (in this case, the silent carrier variant NM_000344.4:c.*3+80T>G)	string
qual	Quality that atleast one copy of the variant allele is found	non-negative floating point number
altCopyNumber	detected copy number of the variant allele	non-negative floating point number
altCopyNumberQuality	Quality of the detected copy number	non-negative floating point number

The variant NM_000344.4:c.*3+80T>G is also reported in a <output-file-prefix>.targeted.vcf [.gz] file in the output directory. The output file is a VCFv4.2 formatted file and possibly compressed. The variant is reported with the VARIANT_IN_HOMOLOGY_REGION flag in the INFO field and also filtered with the TargetedRepeatConflict filter. This variant lies in a region of homology between SMN1 and

SMN2 and hence this variant is reported twice - once for each SMN1 and SMN2 regions - and is connected by the same EVENT in the INFO field. The ploidy of the variant is reported in concordance with the identified genotype.

An example of the vcf entry for the variant NM_000344.4:c.*3+80T>G is as follows.

```
##fileformat=VCFv4.2
...
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT HG04038
chr5 70076654 . T G 150 TargetedRepeatConflict EVENT=NM_000344.4:c.*3+80T>G;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION GT:GQ 0/0/0/1:55
chr5 70952074 . T G 150 TargetedRepeatConflict EVENT=NM_000344.4:c.*3+80T>G;EVENTTYPE=VARIANT_IN_HOMOLOGY_REGION GT:GQ 0/0/0/1:55
...
```

The variant NM_000344.4:c.*3+80T>G in the <output-file-prefix>.targeted.vcf[.gz] file can also be included in the <output-file-prefix>.hard-filtered.vcf[.gz] by including the smn entry int the --targeted-merge-vc list, i.e. --targeted-merge-vc smn. The output file <output-file-prefix>.targeted.vcf[.gz] is compressed by default. This option can be disabled using --enable-vcf-compression=false.

Setting the option --targeted-enable-legacy-output=true the caller also generates an SMN specific output file called <output-file-prefix>.smn.tsv in the output directory, which is deprecated for DRAGEN v4.2. The output file contains a header line followed by a sample call line that contains the following tab-delimited fields.

Field Header	Description	Value(s)
Sample	Sample name	String
isSMA	SMA affected status	True False None*
isCarrier	SMA carrier status	True False None*
SMN1_CN	Copy number of SMN1	Nonnegative integer None*

Field Header	Description	Value(s)
SMN2_CN	Copy number of SMN2	Nonnegative integer None*
SMN2delta7-8_CN	Copy number of SMN2Δ7–8 (deletion of exon 7 and 8)	Nonnegative integer
Total_CN_raw	Raw normalized depth of total SMN	Floating point
Full_length_CN_raw	Raw normalized depth of intact SMN	Floating point
SMN1_CN_raw	Raw SMN1 CN values at differentiating sites	Eight comma-separated floating point values

* The value `None` indicates a confident call could not be made.

The following is an example `<output-file-prefix>.smn.tsv` output file.

```
#Sample  isSMA  isCarrier  SMN1_CN  SMN2_CN  SMN2delta7-8_CN  Total_CN_raw  Full_length_CN_raw  SMN1_CN_raw
HG00111  False  False      2        3        0          4.88       5.04
1.00,2.74,2.10,1.89,2.21,1.63,2.47,2.00
```

Command-Line Examples

To enable the SMN Caller, use `--enable-smn=true` or `--enable-targeted=true` (the option `--enable-targeted=true` also enables other DRAGEN targeted callers). The SMN Caller is disabled by default. The SMN Caller can run from FASTQ input with the mapper enabled or from prealigned BAM/CRAM input. You can also enable the SMN Caller in parallel with any other germline variant callers for a WGS germline analysis workflow. For information on other variant callers, see [DRAGEN DNA Pipeline on page 79](#).

FASTQ Input

The following command-line example uses FASTQ input.

```
dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/8 \
--fastq-file1 /staging/test/data/NA12878_R1.fastq \
--fastq-file2 /staging/test/data/NA12878_R2.fastq \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--RGID DRAGEN_RGID \
```

```
--RGSM NA12878 \
--enable-map-align=true \
--enable-smn=true
```

Prealigned BAM Input

The following command-line example uses BAM input that has already been aligned.

```
dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/8 \
--bam-input /staging/test/data/NA12878.bam \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--enable-map-align=false \
--enable-smn=true
```

¹ ChenX, Sanchis- Juan A, French CE, et al. Spinal muscular atrophy diagnosis and carrier screening from genome sequencing data. *Genetics in Medicine*. 2020;22(5):945-953. doi:10.1038/s41436-020-0754-0

Structural Variant Calling

The DRAGEN Structural Variant (SV) Caller integrates and extends Manta structural variant calling to provide SV and indel calls 50 bases or larger. SVs and indels are called from mapped paired-end sequencing reads. The SV caller is optimized for analysis of diploid germline variation in small sets of individuals .

The SV caller performs the following actions:

- Discovers, assembles, and scores large-scale SVs, medium-sized indels, and large insertions within a single efficient workflow.
- Combines paired and split-read evidence during SV discovery and scoring to improve accuracy, but does not require split-reads or successful breakpoint assemblies to report a variant in cases where there is strong evidence otherwise.
- Scores known SV deletions and insertions from an input VCF file against one or more input samples, either as a standalone procedure or together with standard SV discovery.
- Provides scoring models for:
 - Germline variants in small sets of diploid samples
 - Somatic and germline variants in tumor-only samples

All SV and indel inferences are output in VCF 4.2 format.

DRAGEN SV Caller Overview

The DRAGEN SV Caller divides the SV and indel discovery process into the following steps.

1. Reads input files to estimate alignment statistics, including fragment size distribution and chromosome level depth. For more information on the SV Caller input options, see [Command-Line Options on page 302](#).
2. Scans the genome or a subset of the genome (specified by the call regions) to build various genome-wide data structures, including a breakend association graph of all SV associated regions. The graph contains edges that connect all regions of the genome that have a possible breakend association. Edges can connect two different regions of the genome to represent evidence of a long-range association, or an edge can connect to a region to capture a local indel/small SV association. These associations are more general than a specific SV hypothesis and multiple breakend candidates might be found on one edge. Typically only one or two candidates are found per edge. Instead of passing an inclusion region BED file, an exclusion region BED file can be passed to DRAGEN so that any SV breakend that overlaps with these regions gets removed from downstream analysis. The excluded regions are removed from the graph building process, but active regions can get extended and present in the excluded regions in the refinement step. This can happen for the active regions that are close to the boundaries of the excluded regions. Hence, the final SV calls may still get extended to these regions.
3. Analyzes the breakend association graph to discover candidate SVs, then scores discovered candidate SVs and any known SVs from the input. Analysis and scoring are performed as follows:
 - a. Infers SV candidates that are associated with the given graph edge.
 - b. Assembles the SV breakends.
 - c. Merges discovered SV candidates with any known SV candidates included in the input data.
 - d. Scores/genotypes and filters each SV candidate under various biological models (currently germline, tumor-normal, and tumor-only).
 - e. Outputs scored SVs to VCF.

DRAGEN SV Caller Capabilities

The DRAGEN SV Caller can discover all identifiable structural variant types in the absence of copy number analysis and large-scale *de novo* assembly. For more information on detectable types, see [Detected Variant Classes on page 293](#).

For each structural variant and indel, the SV Caller attempts to assemble the breakends to base pair resolution and report the left-shifted breakend coordinate (per the VCF 4.2 SV reporting guidelines), together with any breakend homology sequence and/or inserted sequence between the breakends. Often the assembly will fail to provide a confident explanation of the data. In this case, the variant is reported as IMPRECISE, and scored according to the paired-end read evidence only.

You can provide known SVs as input for forced genotyping. This known SV input can be scored either standalone or together with the standard SV discovery workflow, in which case the known and discovered SVs are merged.

The sequencing reads provided as input to the SV Caller are expected to be from a paired-end sequencing assay that results in an "innie" orientation between the two reads of each sequence fragment, each presenting a read from the outer edge of the fragment insert inward.

The SV Caller is primarily tested for whole-genome and whole-exome (or other targeted enrichment) sequencing assays on DNA. For these assays the following applications are supported:

- Joint analysis of 5 or fewer diploid individuals
- Subtractive analysis of a matched tumor-normal sample pair
- Analysis of an individual tumor sample

For joint analysis, there is no specific restriction against larger cohorts, but there might be stability or call quality issues.

When performing somatic calling on liquid tumor samples, the matched normal sample might be contaminated with tumor cells. The contamination can substantially reduce somatic variant recall. To account for Tumor-in-Normal (TiN) contamination, you can enable liquid tumor mode. For more information, see [Liquid Tumor Calling on page 302](#).

Tumor samples can be analyzed without a matched normal sample. In this case, both germline and somatic variants are scored and reported in the output.

Modes of Operation

Structural Variant calling can run in the following modes:

- Standalone—Uses mapped BAM/CRAM input files. If you have not mapped and aligned your data yet, see [Input Requirements on page 299](#) This mode requires the following options:
 - `--enable-map-align false`
 - `--enable-sv true`
- Integrated—Automatically runs on the output of the DRAGEN mapper/aligner. This mode requires the following options:
 - `--enable-map-align true`
 - `--enable-sv true`
 - `--enable-map-align-output true`
 - `--output-format bam`

You can also enable Structural Variant calling with any other caller.

The following is an example command line for Integrated mode:

```
dragen -f \
--ref-dir=<HASH_TABLE> \
--enable-map-align true \
```

```
--enable-map-align-output true \
--enable-sv true \
--output-directory <OUT_DIR> \
--output-file-prefix <PREFIX> \
--RGID Illumina_RGID \
--RGSM <sample name> \
-1 <FASTQ1> \
-2 <FASTQ2>
```

The following is an example command line for joint diploid calling in standalone mode:

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

Detected Variant Classes

The SV Caller can discover all variation classes that can be explained as novel DNA adjacencies in the genome. Novel DNA adjacencies are classified into the following categories based on the breakend pattern:

- Deletions
- Insertions. SV insertions can be divided into the following two subclasses depending on if the inserted sequence can be fully assembled.
 - Fully-assembled insertions
 - Partially-assembled (ie. inferred) insertions
- Tandem Duplications
- Unclassified breakend pairs corresponding to intra- and inter-chromosomal translocations, or complex structural variants.

Known Limitations

The SV Caller cannot directly discover the following variant types:

- Dispersed duplications.
 - Dispersed duplications can be indirectly called as insertions or unclassified breakends.

- Most expansion/contraction variants of a reference tandem repeat.
- Breakends corresponding to small inversions.
 - The limiting size is not tested, but in theory detection falls off below ~200 bases. Micro-inversions might be detected indirectly as combined insertion/deletion variants.
- Fully-assembled large insertions.
 - The maximum fully-assembled insertion size should correspond to approximately twice the read-pair fragment size, but power to fully assemble the insertion should fall off to impractical levels before this size.
 - The SV Caller does detect and report very large insertions when the breakend signature of such an event is found, even though the inserted sequence cannot be fully assembled.

More general repeat-based limitations exist for all variant types:

- Power to assemble variants to breakend resolution falls to zero as breakend repeat length approaches the read size.
- Power to detect any breakend falls to (nearly) zero as the breakend repeat length approaches the fragment size.

While the SV Caller classifies certain novel DNA-adjacencies into variant classes, it has a limited ability to infer high-level events resulting from complex rearrangements, so certain calls summarized as deletions, duplications, and insertions might be better described by looking at the full system of breakends and copy number changes associated with a given event.

Forced Genotyping Capability

The DRAGEN SV caller is capable of forced genotyping a set of SVs input from a VCF file. Forced genotyping means that the input SVs are scored and emitted in the output of the SV Caller even if the variant is not supported in the sample data. For example, given a germline analysis, the input variants are processed and written to the output VCF, even if the variant quality falls below the threshold normally required for an SV to be emitted.

Forced genotyping typically enables known SVs to be detected at higher recall than standard SV discovery (particularly for SV discovery on a lower-depth sample). Forced genotyping can also be useful to assert against the presence of an SV allele. For example, you can use forced genotyping to distinguish a confident homozygous reference genotype from a lack of sequencing coverage over the SV locus.

Forced genotyping SVs are processed according to the current SV analysis being run. For example, if a germline analysis is configured by providing one or more normal samples as input, then the input SVs are scored under a germline model.

Forced genotyping alleles are always emitted in the output and might have modified scoring and filtering rules applied compared to SVs only discovered from the sample data.

Forced Genotyping Modes

Forced Genotyping can be run in two modes.

- Standalone—Only SVs described in an input VCF are scored and emitted.
- Integrated—The standard SV discovery analysis is run and the results are merged with SVs scored from the forced genotyping input. The workflow outputs the union of SVs discovered from the sample data and any additional forced genotyping alleles. The workflow is run whenever the `--sv-discovery` option is true.

Forced Genotyping Inputs

You can specify forced genotyping input using the `--sv-forcegt-vcf` option. The input must be a VCF of SV alleles. The SV allele types are restricted to insertions, deletions, tandem duplications, and breakends, which are not labeled with the `INFO/IMPRECISE` flag. The following are the filtering criteria required for the VCF record to be processed as an input SV allele. If any of the criteria are not met, the VCF record is removed from the set of input SVs for forced genotyping. When a forced genotyping VCF is specified on the command line, the SV caller reports the total number of SV records used as input SVs and the total number of records filtered (if any) due to the following criteria.

- Describes an insertion, deletion, tandem duplication, or breakend record.
- Cannot contain the `INFO/IMPRECISE` flag.
- Cannot contain multiple ALT alleles.
- Has a `FILTER` value of `PASS` or unknown (`.`).
- All indels are at least the minimum scored variant size (default is 50).
- Cannot repeat an SV allele previously described in the same file.
- The `REF` field cannot be empty or unknown (`.`).

You must describe insertions using the VCF small indel format including an `ALT` entry that describes the complete insertion sequence. Using `<INS>` as a symbolic alt allele is not accepted. You can describe deletions using either the VCF small indel format or the `` symbolic alt allele. For any variant described using a symbolic alt allele, you must also provide a value for `INFO/END`. Inversions represented in a single VCF record using the `<INV>` alt allele are not accepted, but the inversion can be genotyped if converted to a set of breakend records. Each breakpoint is described by a pair of breakend VCF records. If the forced genotyping input contains just one record of the pair and the input conditions above are met, the input is still accepted for forced genotyping, and the distal breakend is inferred from the local record.

You can describe breakpoint insertions for non-insertion SV alleles using one of the following two methods. Both methods correspond to the format used to describe breakpoint insertions in the SV VCF output.

- For SVs described using the symbolic `ALT` format, such as ``, the `INFO/SVINSSEQ` field is parsed to read the breakpoint insertion sequence.
- For smaller indels described directly in the `REF` and `ALT` fields, the contents of the `ALT` field describe the breakend sequence.

Forced Genotyping Output

Forced genotyping SVs are always output to the standard VCF output of the SV Caller, regardless of whether the forced genotyping is standalone or integrated with SV calling. When the same SV allele is independently discovered from the sample data, only the discovered SV appears in the final output. The discovered SV allele is annotated to indicate the match to a forced genotyping input SV, and the scoring and filtration rules are changed to match.

VCF output records influenced by forced genotyping have the following associated fields.

- The flag `INFO/NotDiscovered` is set for any VCF record that was not independently discovered from the sample data. When forced genotyping is in standalone, all output records contain the flag. When integrated with SV calling, the flag can distinguish the SV alleles that would not have been discovered in a standard SV analysis.
 - For these variants only, the usual SV caller ID field generated from the SV Locus graph is not available, instead the ID is taken from the corresponding user input VCF. The suffix `UserInput${InputVCFRecordNumber}` is appended to the ID, separated by an underscore. If your input VCF contains only one of the two VCF records that comprise a breakend variant, then the ID is taken from the mate breakend record and the `_Mate` suffix is added.
- Any output VCF record that corresponds to a forced genotyping input VCF record has the value `INFO/UserInputId=${ID}` set to reflect the VCF ID value of the input VCF record. The corresponding record might have also been discovered independently from the sample data and might not have the `INFO/NotDiscovered` flag set.
- Any output VCF record that corresponds to a forced genotyping input VCF record containing forced genotyping alleles that match exactly to an input SV has the flag `INFO/KnownSVScoring`. VCF records with this flag are always emitted in the output of the SV Caller. Several filters, such as `MaxDepth`, are not applied.

Systematic Noise Filtering

When DRAGEN-SV is used in the somatic mode (tumor-only or tumor-normal), a BEDPE file with a set of paired-end regions in the BEDPE file format can be specified to filter out sequencing / systematic noise and also recurrent germline calls. Any variant that overlaps with one of the systematic noise paired regions (with a population count of at least 2) and has the same orientation will be marked as `SystematicNoise` in the final VCF file. This BEDPE file can be passed via the command line option `--sv-systematic-noise`.

The systematic noise BEDPE file is built using VCFs that were generated by the DRAGEN-SV tumor-only pipeline when run on a subset of 100 unrelated normal samples from the 1000 Genomes Project. The following prebuilt systematic noise files for WGS are available for download on the [Illumina DRAGEN Bio-IT Platform Product Files](#) page. Each systematic noise file contains a version string that DRAGEN uses to check the compatibility by default and exits early if a wrong systematic noise file is provided.

Pre-built Systematic Noise File	Comment	Systematic Noise Version	DRAGEN Compatibility
WGS_v2.0.0_hg19_sv_systematic_noise.bedpe.gz	>30x coverage using the Illumina NovaSeq 6000 system with 2x150bp reads for the HG19 reference	2.0.0	4.2*
WGS_v2.0.0_hg38_sv_systematic_noise.bedpe.gz	>30x coverage using the Illumina NovaSeq 6000 system with 2x150bp reads for the HG38 reference	2.0.0	4.2*
WGS_v2.0.0_hs37d5_sv_systematic_noise.bedpe.gz	>30x coverage using the Illumina NovaSeq 6000 system with 2x150bp reads for the HS37D5 reference	2.0.0	4.2*

The systematic noise BEDPE should follow a particular format

ID	Description
contig1	chromosome of the first region (string)
start1	start position of the first region (0-based left-most position of the first breakpoint containing genomic interval, integer)
end1	end position of the first region (0-based left-most position of the first breakpoint containing genomic interval, integer)
contig2	chromosome of the second region (string)
start2	start position of the second region (0-based left-most position of the second breakpoint containing genomic interval, integer)
end2	end position of the second region (0-based left-most position of the second breakpoint containing genomic interval, integer)
event_id	The paired region unique ID (string)
score	The number of occurrences in the cohort

ID	Description
orientation1	direction of breakpoint1 relative to the reference; "+" indicates to the right, "-" to the left (string, "+", or "-")
orientation2	direction of breakpoint2 relative to the reference; "+" indicates to the right, "-" to the left (string, "+", or "-")

SV Scoring

The SV caller applies a diploid scoring model for one or more diploid samples(treated as unrelated), as well as a somatic scoring model when a tumor and matched normal sample pair are given.

Germline scoring model

The germline scoring model produces diploid genotype probabilities for each candidate structural variant. Most candidates are approximated as independent for scoring purposes and modeled under a bi-allelic and diploid genotype likelihood setting.

DRAGEN solves for the posterior probability over possible genotypes, given the sequencing data, by approximating it proportionally to the product between the prior probability of a genotype values and the conditional probability of observing a set of read fragments(post-filtering), given the underlying genotype.

DRAGEN treats each read fragment independently and represents the conditional probability of the set of read fragments as the product over all the individual read fragments'. For each individual read fragment's conditional probability, DRAGEN combines both paired-read and split-read evidence components, and approximates their contributions as independent by representing it as a product of these components, with the condition that the paired-read component is weighted by a linear ramp from one to zero depending on the candidate event type and size as tiny event will not affect pair-read mapping status significantly.

- The paired read component is modeled as a function measuring the deviation of the inferred fragment length from the overall distribution.
- The split-read component is modeled as a function measuring the correctness of a read alignment to the breakend by multiplying across all the non-gap bases' probability of observing a certain base call given the corresponding base of the evaluated allele.

Each read fragment may contribute only paired read support, only split-read support, or both. Where a fragment contributes split-read support, this support may come from either or both reads in the read pair.

Somatic scoring model

The somatic scoring model is a Bayesian probabilistic model using a tumor-normal joint genotyping approach. It aims to call somatic structural variants in tumors while avoiding germline variants and noisy variants. The tumor and normal allele frequencies are treated as nonindependent random variables. DRAGEN calculates posterior probabilities for a range of genotype hypotheses, under the assumption that the normal sample conforms to the diploid germline genotype considering homozygous reference, heterozygous, and homozygous states. The tumor sample is a mixture of the germline genotype and somatic allele. For the somatic genotype, only two states are considered when referring to the absence and presence of the somatic variant in the tumor sample. In cases where the somatic variant is not present, unsystematic independent noise is accounted for, while assuming an error-free scenario when the somatic variant is considered. To calculate the genotype likelihoods, the model integrates allele frequency likelihoods over the joint tumor and normal allele frequencies, and applies modifications to address liquid tumors with Tumor-in-Normal (TiN) contamination. The integration is approximated with a discrete summation. In these calculations, the likelihood for each read to support a given allele is shared with the germline scoring model. The tumor-only somatic scoring model is seen as a special case of the somatic scoring model in the absence of normal data (zero coverage). The posterior probability is converted into a Phred quality score and reported in the VCF output INFO/SOMATICSCORE field.

Input Requirements

When running the SV Caller, the input sequencing reads must be from a standard Illumina paired-end sequencing assay with an FR read pair orientation, where for each sequence fragment, a read proceeds from each end of the fragment inwards. For more information, see [DRAGEN SV Caller Capabilities on page 291](#).

The SV Caller is optimized for paired-end libraries where the fragment size is typically larger than the size of both reads. Overlapping read pairs can be used to discover SVs, but might not always be handled optimally. For libraries where the typical fragment size is less than the read length, the SV caller attempts to differentiate reads sequencing into adapter sequence from the variant signal. In such cases, the SV Caller's input quality checks may fail and cause SV analysis to be skipped.

If using the standalone mode, your BAM/CRAM inputs must first be mapped. If you have not mapped and aligned your data yet, you can generate an alignment file.

Alignment Contig Checks

If running from a mapped and aligned BAM, then the contigs specified in the header must strictly match those in the DRAGEN hashtable specified in the current analysis. Missing or extra contigs will lead to a "Reference genome mismatch" configuration error and the analysis will not proceed. If such an error is observed, it is recommended to regenerate the alignment file with the intended DRAGEN hashtable, or to run with the DRAGEN map/align module enabled.

Input Quality Checks

The SV Caller runs quality checks on the input sequencing reads for each sample to make sure that the input corresponds to a paired read assay with the expected FR orientation, before estimating the fragment size distribution. To check consensus read pair orientation, a subset of high-quality read pairs is sampled. At least 90% of these must have the expected FR orientation for SV analysis to continue, otherwise the SV caller issues a warning, skips any further analysis, and the resulting output files display empty results.

The SV Caller can tolerate nonpaired reads in the input, if sufficient paired-end reads exist to estimate the fragment size distribution. To estimate the fragment size distribution, the SV Caller requires at least 100 read pairs that meet the quality requirements of the estimation routine. Both reads of the pair must have a non-zero mapping quality to the same chromosome, are not filtered or part of a split read mapping, and do not contain indels or soft-clipping. If a sample does not contain a sufficient number of such read pairs, the SV Caller issues a warning, skips any further analysis, and writes empty results to its output files.

Read Groups

The SV Caller disregards any read group labels applied to the input sequences. Each input sample is treated as a separate library with a single fragment size distribution.

File Format

In standalone mode, input sequencing reads must be mapped and provided as input in either BAM or CRAM format. Each input file must be coordinate sorted and indexed to produce an htseq-style index in a file named to match the input BAM or CRAM file with an additional *.bai, *.crai, or *.csi file name extension. For more information on standalone mode, see [Modes of Operation on page 292](#).

At least one BAM or CRAM file must be provided for the normal or tumor sample. A matched tumor-normal sample pair can be provided as well. If multiple input files are provided for the normal sample, each file is treated as a separate sample as part of a joint diploid sample analysis.

In standalone mode, input BAM or CRAM files contain the following limitations:

- Alignments cannot have an unknown read sequence (SEQ="")
- Alignments cannot contain the "=" character in the SEQ field.
- Alignments cannot use the sequence match/mismatch ("=/X"). CIGAR notation RG (read group) tags in the alignment records are ignored. Each alignment file is treated as representing one sample.
- Alignments with base call quality values greater than 70 are rejected. These are not supported on the assumption that this indicates an offset error.

Generate an Alignment File

The following command-line examples show how to run the DRAGEN map/align pipeline depending on your input type. The map/align pipeline generates an alignment file in the form of a BAM or CRAM file that can then be used in the pipeline.

You need to generate alignment files for all samples that have not already been mapped and aligned. Each sample must have a unique sample identifier. Use the `--RGSM` option to specify the identifier. For BAM and CRAM input files, the sample identifier is taken from the file, so the `--RGSM` option is not required.

The following example command maps and aligns a FASTQ file:

```
dragen \
-r <HASHTABLE> \
-1 <FASTQ1> \
-2 <FASTQ2> \
--RGSM <SAMPLE> \
--RGID <RGID> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align true
```

The following example command maps and aligns an existing BAM file:

```
dragen \
-r <HASHTABLE> \
--bam-input <BAM> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align true
```

The following example command maps and aligns an existing CRAM file:

```
dragen \
-r <HASHTABLE> \
--cram-input <CRAM> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-map-align true
```

Exome/Targeted Calling

The SV caller can be configured for targeted sequencing inputs, which disables high-depth filters. Exome mode can be directly set to true or false with the command line option `--sv-exome`. If not directly set, exome mode defaults to false unless you run the SV caller in integrated mode and there is not more than 50 Gb of sequencing input.

Internal Tandem Duplications Calling

You can use the `--sv-somatic-ins-tandup-hotspot-regions-bed ${BEDFILE}` option to specify ITD hotspot regions to increase sensitivity for calling ITDs in somatic variant analysis. By default, DRAGEN SV automatically selects a reference-specific hotspots BED file from `/opt/edico/config/sv_somatic_ins_tandup_hotspot_*.bed`. The file includes FLT3, ARHGEF7, and KMT2A. To disable this feature, enter `--sv-enable-somatic-ins-tandup-hotspot-regions false`.

Liquid Tumor Calling

Liquid tumors in liquid tumor mode refer to hematological cancers, such as leukemia. In tumor-normal analysis, DRAGEN accounts for Tumor-in-Normal (TiN) contamination by running liquid tumor mode. You can use liquid tumor mode to account for TiN contamination by allowing a nonzero variant allele frequency for the matched normal when calculating the posterior probability of the somatic state. If the contamination is not accounted for, it can severely impact sensitivity by suppressing true somatic variants.

Use the following two options to control liquid tumor mode behavior.

- `--sv-enable-liquid-tumor-mode`—Enable liquid tumor mode. Liquid tumor mode is disabled by default.
- `--sv-tin-contam-tolerance`—Set the TiN contamination tolerance level. DRAGEN calls variants in the presence of TiN contamination up to a specified maximum tolerance level. You can enter any value between 0–1. The default maximum TiN contamination tolerance is 0.15. If using the default value, somatic variants are expected to be observed in the normal sample with allele frequencies up to 15% of the corresponding allele in the tumor sample.

Command-Line Options

The following command line options are supported for the Structural Variant Caller.

Input and Output Options

The DRAGEN SV pipeline shares the following input and output options with the DRAGEN Host Software. You can use BAM and CRAM files as input. Alternatively, if using read mapping with the SV calling in a single run, you can use all of the DRAGEN input options, including FASTQ, BAM, and CRAM files.

Option	Description
--bam-input	The BAM file to be processed.
--tumor-bam-input	If performing tumor-normal or tumor-only analysis, the tumor BAM file to be processed.
--cram-input	The CRAM file to be processed.
--tumor-cram-input	If performing tumor-normal or tumor-only analysis, the tumor CRAM file to be processed.
--enable-map-align	Enables DRAGEN map/align. The default is true, so all input reads are remapped and aligned unless the option is set to false.
--fastq-file1, --fastq-file2, --fastq-list	Input FASTQ files or list of files to be processed.
--tumor-fastq1, --tumor-fastq2, --tumor-fastq-list	Input tumor FASTQ file or list of files to be processed.
--ref-dir	The DRAGEN reference genome hashtable directory. For more information about the reference genome hashtable, see Prepare a Reference Genome on page 11 .
--output-directory	Output directory where all results are stored.
--output-file-prefix	Output file prefix that will be prepended to all result file names.

SV Calling Options

Option	Description
--enable-sv	Enable or disable the structural variant caller. The default is false.
--sv-call-regions-bed	Specifies a BED file containing the set of regions to call. Optionally, you can compress the file in gzip or bgzip format.
--sv-exclusion-bed	Specifies a BED file containing the set of regions to exclude for the SV calling. Optionally, you can compress the file in gzip or bgzip format.
--sv-region	Limit the analysis to a specified region of the genome for debugging purposes. This option can be specified multiple times to build a list of regions. The value must be in the format "chr:startPos-endPos".

Option	Description
--sv-exome	Set to true to configure the variant caller for targeted sequencing inputs, which includes disabling high depth filters. In integrated mode, the default is to autodetect targeted sequencing input, and in standalone mode the default is false.
--sv-output-contigs	Set to true to have assembled contig sequences output in a VCF file. The default is false.
--sv-forcegt-vcf	Specify a VCF of structural variants for forced genotyping. The variants are scored and emitted in the output VCF even if not found in the sample data. The variants are merged with any additional variants discovered directly from the sample data.
--sv-discovery	Enable SV discovery. This flag can be set to false only when --sv-forcegt-vcf is used. When set to false, SV discovery is disabled and only the forced genotyping input variants are processed. The default is true.
--sv-use-overlap-pair-evidence	Allow overlapping read pairs to be considered as evidence. The default is false.
--sv-somatic-ins-tandup-hotspot-regions-bed	Specify a BED of ITD hotspot regions to increase sensitivity for calling ITDs in somatic variant analysis. By default, DRAGEN SV automatically selects a reference-specific hotspots BED file from /opt/edico/config/sv_somatic_ins_tandup_hotspot_* .bed.
--sv-enable-somatic-ins-tandup-hotspot-regions	Enable or disable the ITD hotspot region input. The default is true in somatic variant analysis.
--sv-enable-liquid-tumor-mode	Enable liquid tumor mode. See Liquid Tumor Calling on page 302 for more information.
--sv-tin-contam-tolerance	Set the Tumor-in-Normal (TiN) contamination tolerance level. See Liquid Tumor Calling on page 302 for more information.
--sv-systematic-noise	Systematic noise BEDPE file containing the set of noisy paired regions (optionally gzip or bzip compressed). For more information see Systematic Noise Filtering for more information.
--sv-min-edge-observations	Remove all edges from the graph with less than this many observations. The default value is set to 3.
--sv-min-candidate-spanning-count	Run SV caller and report all large SVs with at least this many spanning support observations. The default value is set to 3.

Option	Description
--sv-min-candidate-variant-size	Run SV caller and report all SVs/indels at or above this size. The default value is set to 10.
--sv-min-scored-variant-size	After candidate identification, only score and report SVs/indels at or above this size. The default value is set to 50. This parameter doesn't affect the somatic hotspot region.
--sv-hotspot-min-scored-variant-size	After candidate identification, only score and report SVs/indels at or above this size inside the SV hotspot region, which includes FLT3, ARHGEF7, and KMT2A genes by default. The default value is set to 25.

Structural Variant VCF Output

The structural variants VCF output file is available in the output directory. The file is named <output-file-prefix>.sv.vcf.gz. The contents of the file depend on the type of analysis.

For each major analysis category (germline, tumor-normal, and tumor-only), the appropriate VCF output file is output, reflecting variant calls made under the variant calling mode corresponding to the given analysis type.

The Structural Variant caller produces additional output in the <output-directory>/sv/ directory. The <output-directory>/sv/results subdirectory contains additional variants and statistics output files. Additional subdirectories contain logs and intermediate outputs from the variant calling process. The intermediate output files in the <output-directory>/sv/workspace subdirectory are subject to removal or changes in the future and should not be treated as standard output files.

Structural Variant Predictions

In <output-directory>/sv/results/variants, the SV Caller outputs a set of VCF files. Two VCF files are created for a germline analysis. For tumor-normal analysis, an additional somatic VCF is produced for a tumor-normal subtraction. For tumor-only analysis, the SV Caller produces an additional output VCF. The following VCF files are output:

File	Description
diploidSV.vcf.gz	SVs and indels scored and genotyped under a diploid model for the set of samples in a joint diploid sample analysis or for the normal sample in a tumor-normal subtraction analysis. In the case of a tumor-normal subtraction, the scores in this file do not reflect any information from the tumor sample.

File	Description
somaticSV.vcf.gz	SVs and indels scored under a somatic variant model. This file is only produced if a tumor sample alignment file is supplied during configuration
candidateSV.vcf.gz	Unscored SV and indel candidates. Only a minimal amount of supporting evidence is required for a variant to be entered as a candidate in this file. A variant must be a candidate to be considered for scoring, therefore a variant does not appear in the other VCF outputs if it is not present in the file. The file includes indels of size 8 and larger. The smallest indels are provided for workflow flexibility, but are not scored. Indel scoring starts at size 50.

For tumor-only analysis, the SV Caller produces the following additional output VCF:

File	Description
tumorSV.vcf.gz	A subset of the candidateSV.vcf.gz file after removing redundant candidates and small indels less than the minimum scored variant size (50). The SVs are scored, and a subset of the filters from the scored tumor-normal model is applied to the single tumor case to improve precision.

VCF Output

VCF output follows the VCF 4.2 specification for describing structural variants. It uses standard field names wherever possible. All custom fields are described in the VCF header. The following sections provide information on the variant representation details and the primary VCF field values.

VCF Sample Names

Sample names output in the VCF output are extracted from each input alignment file from the first read group (@RG) record found in the header. If no sample name is found, a default (SAMPLE1, SAMPLE2, etc.) label is used instead.

Small Indel Representation

All variants are reported in the VCF using symbolic alleles unless they are classified as a small indel, in which case full sequences are provided for the VCF REF and ALT allele fields. A variant is classified as a small indel if all of the following criteria are met:

- The variant can be entirely expressed as a combination of inserted and deleted sequences.
- The deletion or insertion length is not 1000 or greater.

- The variant breakends and/or the inserted sequence are not imprecise.
- The variant has not been converted from a deletion to intra-chromosomal breakends by the depth-based SV classification routine.

When VCF records are output in the small indel format, they also include the CIGAR INFO tag describing the combined insertion and deletion event.

Insertions with Incomplete Insert Sequence Assembly

Large insertions are reported in some cases even when the insert sequence cannot be fully assembled. In this case, the SV Caller reports the insertion using the <INS> symbolic allele and includes the special INFO fields LEFT_SVINSSEQ and RIGHT_SVINSSEQ to describe the assembled left and right ends of the insert sequence. The following is an example of such a record from the joint diploid analysis of NA12878, NA12891 and NA12892 mapped to hg19:

```
chr1 11830208 MantaINS:1577:0:0:0:3:0 T <INS> 999 PASS
END=11830208;SVTYPE=INS;CIPOS=0,12;CIEND=0,12;HOMLEN=12;HOMSEQ=TAAATTTTCTT;LEFT_
SVINSSEQ=TAAATTTCTTTCTTTTTAAATTATTTTATTGATAATTCTGGGTGTTCTCACAGA
GGGGGATTGGCAGGGTCACGGGACAACAGTGGAGGGAAAGGTCAAGCAGACAAACAAGTGAACAAAGGTCTGGTTTC
CCAGGCAGAGGCCCTGCAGCTCCGAGTGGTGTCCCTGATTACCTGAGATTAGGATTGTGATGACTCCAA
CGAGCATGCTGCCTCAAGCATCTGTTCAACAAAGCACATCTGCAGTGACTGCCCTTAATTCAACCCGAGTGGACAC
AGCACATGTTCAAAGAG;RIGHT_
SVINSSEQ=GGGGCAGAGGCCTCCCCACATCTCAGATGATGGCGGCCAGGCAGAGACGCTCCTCAGTCAGATGT
GATGGCGGCTGGAAAGAGGCCTCCACTTCCTAGATGGACGGCGGCCGGAGACGCTCCTCAGTCAGACT
GGGCAGCCAGGCAGAGGGGCTCCTCACATCCCAGACGATGGCGGCCAGGCAGAGACACTCCCAGACGGGG
TGGCGGCCGGCAGAGGCTGCAATCTGGCACTTGGAGGCCAAGGCAGGCAGTGCTCCTGCCCTGGGCCCGCG
GGGCCGTCCGCTCCAGCCGCTGCCCTCC
GT:FT:GQ:PL:PR:SR
0/1:PASS:999:999,0,999:22,24:22,32 0/1:PASS:999:999,0,999:18,25:24,20
0/0:PASS:230:0,180,999:39,0:34,0
```

Normalizing Small Tandem Duplications

The SV caller can also represent tandem duplications as insertions. This representation creates ambiguity in how the variants are presented in the VCF output, especially for small tandem duplications. The representation can lead to complications, such as unrecognized call duplication.

To better normalize the SV caller output, so that the same variant type is not represented in two different VCF formats, small tandem duplications (< 1000 bases) are converted to insertions in the VCF output. Insertions converted from such tandem duplications have a formatting similar to incomplete insertions, using the symbolic allele <INS> for the ALT field. The following example shows an insertion, which was converted from a tandem duplication during this normalization process.

```
chr2 2520057 MantaDUP:TANDEM:53645:0:1:0:0:0 T <INS> 813 PASS
END=2520057;SVTYPE=INS;SVLEN=52;DUPSLEN=52 GT:FT:GQ:PL:PR:SR
0/1:PASS:393:863,0,390:25,0:19,25
```

Converted insertions include copies of certain output fields. The fields appear the same as in a tandem duplication record. For example, `INFO/DUPSVINSSEQ` provides a copy of the breakpoint insertion value computed for the duplication. In the context of a duplication, the breakpoint insertion value would normally be written to `INFO/SVINSSEQ`. The following example shows a converted insertion with a breakpoint insertion value:

```
chr2 2645730 MantaDUP:TANDEM:53649:0:1:0:0:0 C <INS> 367 PASS
END=2645730;SVTYPE=INS;SVLEN=97;DUPSLEN=86;DUPSVINSLEN=11;DUPSVINSSEQ=CTCACCTT
CAT GT:FT:GQ:PL:PR:SR 0/1:PASS:367:417,0,386:19,0:20,15
```

For more information about copied `INFO` fields, see [VCF INFO Fields on page 311](#). All `INFO` fields use the same `DUP` prefix.

Inversions

Inversions are reported as a set of breakends. For example, given a simple reciprocal inversion, four breakends are reported, sharing the same `EVENT` `INFO` tag. The following is an example breakend records representing a simple reciprocal inversion:

```
chr1    17124941    MantaBND:1445:0:1:1:3:0:0    T    [chr1:234919886
[T    999    PASS    SVTYPE=BND;MATEID=MantaBND:1445:0:1:1:3:0:1;CIPOS=0,1;HOMLEN=1;HOMSEQ=T
;EVENT=MantaBND:1445:0:1:0:0:0:0;JUNCTION_QUAL=254;BND_DEPTH=107;MATE_BND_
DEPTH=100    GT:FT:GQ:PL:PR:SR    0/1:PASS:999:999,0,999:65,8:15,51

chr1    17124948    MantaBND:1445:0:1:0:0:0:0    T    T]chr1:234919824]    999    PASS    SV
TYPE=BND;MATEID=MantaBND:1445:0:1:0:0:0:1;EVENT=MantaBND:1445:0:1:0:0:0:0;JUNCTION_
QUAL=999;BND_DEPTH=109;MATE_BND_
DEPTH=83    GT:FT:GQ:PL:PR:SR    0/1:PASS:999:999,0,999:60,2:0,46

chr1    234919824    MantaBND:1445:0:1:0:0:0:1    G    G]chr1:17124948]    999    PASS    SV
TYPE=BND;MATEID=MantaBND:1445:0:1:0:0:0:0;EVENT=MantaBND:1445:0:1:0:0:0:0;JUNCTION_
QUAL=999;BND_DEPTH=83;MATE_BND_
DEPTH=109    GT:FT:GQ:PL:PR:SR    0/1:PASS:999:999,0,999:60,2:0,46

chr1    234919885    MantaBND:1445:0:1:1:3:0:1    A    [chr1:17124942
[A    999    PASS    SVTYPE=BND;MATEID=MantaBND:1445:0:1:1:3:0:0;CIPOS=0,1;HOMLEN=1;HOMSEQ=A
;EVENT=MantaBND:1445:0:1:0:0:0:0;JUNCTION_QUAL=254;BND_DEPTH=100;MATE_BND_
DEPTH=107    GT:FT:GQ:PL:PR:SR    0/1:PASS:999:999,0,999:65,8:15,51
```

Depth-Based SV Type Classification

In the germline calling model, when SV candidates are discovered from the sample data and have sufficient paired and split read evidence to be reported in the output, the SV caller applies additional depth-based tests to more accurately classify certain SV candidate types. Candidate breakpoints that are consistent with a deletion are tested for the lower read depth that is expected inside the deleted region. Candidate breakpoints consistent with a tandem duplication are tested for the higher read depth expected in the duplicated region. Candidate SV calls that fail the depth-based tests are still reported in the output, but changed to intrachromosomal breakends. Candidate SV calls that pass continue to be reported in the standard deletion and tandem duplication output formats.

SV Breakpoints

SV Breakpoint Insertions

SVs frequently include a small sequence insertion at the breakpoint. Breakpoint insertions are represented differently depending on the SV type. The `INFO/SVINSSEQ` field in the VCF output provides the most general description of breakpoint insertions by describing the insertion sequence itself. The corresponding `INFO/SVINSLEN` field describes the length of the insertion sequence. For example, the following VCF record describes a large (~8.8 kb) deletion, which includes a single base insertion (C) between the left and right deletion breakends.

```
chr22 17770350 MantaDEL:101:0:1:0:0:0 C <DEL> 687 PASS
END=17779108;SVTYPE=DEL;SVLEN=-8758;SVINSLEN=1;SVINSSEQ=C GT:FT:GQ:PL:PR:SR
0/1:PASS:687:737,0,858:39,20:32,8
```

The `INFO/SVINSSEQ` field is also used to describe breakpoint insertions for tandem duplication and breakend records. The field can also be used to describe the insertion sequence of a large SV insertion. Breakpoint insertions are represented differently in the VCF small indel format. The SV caller represents small deletions and insertions using the VCF small indel format instead of symbolic ALT alleles. Any breakpoint insertion that occurs in the VCF small indel format is represented as part of the VCF ALT field. See [Small Indel Representation on page 306](#) for information on the conditions this format is used for SVs under.

In the following small indel format example, the VCF record describes a 57 base deletion that includes a single base insertion (A) between the left and right deletion breakends.

```
chr22 32981929 MantaDEL:1136:0:0:0:0
TGTATACATATGTGTATACGTATATGTATATGTATGTATACGTATATG TA 537 PASS
END=32981986;SVTYPE=DEL;SVLEN=-57;CIGAR=1M1I57D GT:FT:GQ:PL:PR:SR
0/1:PASS:308:587,0,305:8,0:23,15
```

Breakend records include an additional encoding of breakpoint insertion sequence, as described in the VCF specification for the breakend `ALT` field. The SV caller also provides the information to the `INFO/SVINSSEQ` field for consistency with other SV record types.

The following example shows a breakend connecting a region of chromosomes 1 and 12 in the sample with a breakend insertion sequence of CA between the two breakends. The insertion sequence is described in both the ALT and INFO/SVINNSEQ fields.

```
1 39604587 MantaBND:31780:1:3:0:0:0:1 T TCA[12:6472102[ 774 PASS
SVTYPE=BND;MATEID=MantaBND:31780:1:3:0:0:0:0;SVINSLEN=2;SVINSSEQ=CA;BND_
DEPTH=67;MATE_BND_DEPTH=55 GT:FT:GQ:PL:PR:SR 0/1:PASS:774:824,0,999:63,3:36,33
12 6472102 MantaBND:31780:1:3:0:0:0:0 G ]1:39604587]CAG 774 PASS
SVTYPE=BND;MATEID=MantaBND:31780:1:3:0:0:0:1;SVINSLEN=2;SVINSSEQ=CA;BND_
DEPTH=55;MATE_BND_DEPTH=67 GT:FT:GQ:PL:PR:SR 0/1:PASS:774:824,0,999:63,3:36,33
```

SV Breakpoint Insertion Orientation

The breakpoint insertion sequence is always provided with respect to the strand of the current SV record. Some breakend records have inverted orientation. For inverted orientations, the pair of breakend records contains an insertion sequence that is reverse complemented compared to the mated record.

The following breakend pair example demonstrates an inverted orientation.

```
1 210891730 MantaBND:43882:0:2:0:2:0:1 A AATG]19:45732595] 999 PASS
SVTYPE=BND;MATEID=MantaBND:43882:0:2:0:2:0:0;SVINSLEN=3;SVINSSEQ=ATG;BND_
DEPTH=76;MATE_BND_DEPTH=106 GT:FT:GQ:PL:PR:SR
0/1:PASS:999:999,0,999:69,16:43,55
19 45732595 MantaBND:43882:0:2:0:2:0:0 G GCAT]1:210891730] 999 PASS
SVTYPE=BND;MATEID=MantaBND:43882:0:2:0:2:0:1;SVINSLEN=3;SVINSSEQ=CAT;BND_
DEPTH=106;MATE_BND_DEPTH=76 GT:FT:GQ:PL:PR:SR
0/1:PASS:999:999,0,999:69,16:43,55
```

SV Breakpoint Homology

Each VCF record output by the SV caller is shifted to the left-most position of the exact homology range of the breakpoint. The exact homology range of the breakpoint is the continuous range of positions over which the SV could be represented while still describing the same SV haplotype. The exact homology range is described in the VCF output with the INFO/HOMSEQ field, which describes the sequence of the exact homology range and the corresponding INFO/HOMLEN field, which describes the length of the range.

The following example shows a 62 base deletion with an 11 base breakend homology region. Without left-shifting, the SV has an equivalent representation anywhere from position 39497639 to 39497650.

```
chr22 39497639 MantaDEL:34:85:85:1:0:0
GGGGGGTGGGGCGGGTTGGAGGAGGTTGGCGGGGGCGGGGTTGGAGGAGGTTGGCA G 187 PASS
END=39497701;SVTYPE=DEL;SVLEN=-
```

```
62;CIGAR=1M62D;CIPOS=0,11;HOMLEN=11;HOMSEQ=GGGGGTGGGGG GT:FT:GQ:PL:PR:SR
0/1:PASS:12:237,0,8:4,0:2,8
```

The following examples illustrate simplified exact breakend homology. The example displays one three base deletion and another three base insertion. In both the insertion and deletion, the variant is left-shifted, so that the corresponding VCF record position is 2.

Deletion

Reference: GT**C**AGCGA

Variant: GT---**CGA**

Insertion

Reference: GT---**CAG**

Variant: GT**C**GGCAA

In both the insertion and deletion, there is a single base of exact breakend homology C, so that the same variant can be represented one base to the right.

VCF INFO Fields

ID	Description
IMPRECISE	Flag indicating that the structural variation is imprecise, ie, the exact breakpoint location is not found
SVTYPE	Type of structural variant
SVLEN	Difference in length between REF and ALT alleles
END	End position of the variant described in this record
CIPOS	Confidence interval around POS
CIEND	Confidence interval around END
CIGAR	CIGAR alignment for each alternate indel allele

ID	Description
MATEID	ID of mate breakend
EVENT	ID of event associated to breakend
HOMLEN	Length of base pair identical homology at event breakpoints
HOMSEQ	Sequence of base pair identical homology at event breakpoints
SVINSLEN	Length of insertion
SVINSSEQ	Sequence of insertion
LEFT_SVINSSEQ	Known left side of insertion for an insertion of unknown length
RIGHT_SVINSSEQ	Known right side of insertion for an insertion of unknown length
PAIR_COUNT	Read pairs supporting this variant where both reads are confidently mapped
BND_PAIR_COUNT	Confidently mapped reads supporting this variant at this breakend (mapping may not be confident at remote breakend)
UPSTREAM_PAIR_COUNT	Confidently mapped reads supporting this variant at the upstream breakend (mapping may not be confident at downstream breakend)
DOWNSTREAM_PAIR_COUNT	Confidently mapped reads supporting this variant at this downstream breakend (mapping may not be confident at upstream breakend)
BND_DEPTH	Read depth at local translocation breakend
MATE_BND_DEPTH	Read depth at remote translocation mate breakend
JUNCTION_QUAL	If the SV junction is part of an EVENT (ie, a multi-adjacency variant), this field provides the QUAL value for the adjacency in question only
SOMATIC	Flag indicating a somatic variant
SOMATICSCORE	Somatic variant quality score
SOMATIC_EVENT	If the probability of the SV being a germline variant is greater than the probability of the SV being a somatic variant, this is 0. Otherwise, this is 1.
JUNCTION_SOMATICSCORE	If the SV junction is part of an EVENT (ie, a multi-adjacency variant), this field provides the SOMATICSCORE value for the adjacency in question only
CONTIG	Assembled contig sequence, if the variant is not imprecise (with --outputContig)
DUPSVLEN	Length of duplicated reference sequence
DUPHOMLEN	Length of base pair identical homology at event breakpoints excluding duplicated reference sequence

ID	Description
DUPHOMSEQ	Sequence of base pair identical homology at event breakpoints excluding duplicated reference sequence
DUPSVINSLEN	Length of inserted sequence after duplicated reference sequence
DUPSVINSSEQ	Inserted sequence after duplicated reference sequence
NotDiscovered	Variant candidate specified by the user and not discovered from input sequencing data
UserInputId	Variant ID from user input VCF
KnownSVScoring	Variant is associated with a user specified input variant, therefore scoring and filtration criteria are relaxed under a stronger prior assumption of truth

VCF FORMAT Fields

ID	Description
GT	Genotype
FT	Sample filter. <code>PASS</code> indicates that all filters have passed for this sample
GQ	Genotype Quality
PL	Normalized, Phred-scaled likelihoods for genotypes as defined in the VCF specification
PR	Number of spanning read pairs which strongly (Q30) support the REF or ALT alleles
SR	Number of split-reads which strongly (Q30) support the REF or ALT alleles

VCF FILTER Fields

Germline

The following table lists the VCF FILTER fields applied to germline VCF output.

ID	Level	Description
MinQUAL	Record	QUAL score is less than a threshold. The filter is not applied to records with KnownSVScoring flag.
Ploidy	Record	For DEL and DUP variants, the genotypes of overlapping variants with similar size are inconsistent with diploid expectation. The filter is not applied to records with KnownSVScoring flag.

ID	Level	Description
MaxDepth	Record	Depth is greater than 3x the median chromosome depth near one or both variant breakends. The filter is not applied to records with KnownSVScoring flag.
MaxMQ0Frac	Record	For a small variant (<1000 bases), the fraction of reads in all samples with MAPQ0 around either breakend that exceeds 0.4. The filter is not applied to records with KnownSVScoring flag.
NoPairSupport	Record	For variants significantly larger than the paired read fragment size, no paired reads support the alternate allele in any sample. The filter is not applied to records with KnownSVScoring flag.
SampleFT	Record	No sample passes all the sample-level filters.
MinGQ	Sample	GQ score is less than 15. The filter is applied at sample level and not applied to records with KnownSVScoring flag.
HomRef	Sample	Homozygous reference call. The filter is applied at the sample level.

Tumor- Normal Somatic

The following table lists the VCF FILTER fields applied to tumor-normal somatic VCF output.

ID	Level	Description
MinSomaticScore	Record	SOMATICSCORE is less than a threshold.
MaxDepth	Record	Normal sample site depth is greater than 3x the median chromosome depth near one or both variant breakends. The filter is not applied to records with KnownSVScoring flag.
MaxMQ0Frac	Record	For a small variant (< 1000 bases) in the normal sample, the fraction of reads with MAPQ0 around either breakend exceeds 0.4. The filter is not applied to records with KnownSVScoring flag.
SystematicNoise	Record	Variant overlaps with one of the paired regions in the systematic noise BEDPE file with matched orientation. The filter is not applied to records with the KnownSVScoring flag.

Tumor- Only

The following table lists the VCF FILTER fields applied to tumor-only VCF output.

ID	Level	Description
MinSomaticScore	Record	SOMATICSCORE is less than a threshold.
SystematicNoise	Record	Variant overlaps with one of the paired regions in the systematic noise BEDPE file with matched orientation. The filter is not applied to records with the KnownSVScoring flag.
MaxDepth	Record	Normal sample site depth is greater than 3x the median chromosome depth near one or both variant breakends. The filter is not applied to records with KnownSVScoring flag.
MaxMQ0Frac	Record	For a small variant (<1000 bases), the fraction of reads with MAPQ0 around either breakend exceeds 0.4. The filter is not applied to records with KnownSVScoring flag.

Interpretation of VCF Filters

There are two levels of VCF filters: record level (FILTER) and sample level (FORMAT/FT).

Most record-level filters are independent of sample-level filters. However, in a germline analysis, if none of the samples pass all sample-level filters, the SampleFT filter record-level filter is applied.

Interpretation of INFO/EVENT Field

Some structural variants reported in the VCF, such as translocations, represent a single novel sequence junction in the sample. The INFO/EVENT field indicates that two or more such junctions are hypothesized to occur together as part of a single variant event. All individual variant records belonging to the same event share the same INFO/EVENT string. Note that although such an inference could be applied after SV calling by analyzing the relative distance and orientation of the called variant breakpoints, the SV Caller incorporates this event mechanism into the calling process to increase sensitivity towards such larger-scale events. Given that at least one junction in the event has already passed standard variant candidacy thresholds, sensitivity is improved by lowering the evidence thresholds for additional junctions which occur in a pattern consistent with a multijunction event (such as a reciprocal translocation pair).

Although this mechanism could generalize to events including an arbitrary number of junctions, it is currently limited to two. Thus, at present it is most useful for identifying and improving sensitivity towards reciprocal translocation pairs.

VCF ID Field

The VCF ID, or identifier, field can be used for annotation, or in the case of BND (breakend) records for translocations, the ID value is used to link breakend mates or partners. The following is an example of a VCF ID field from the SV caller.

```
MantaINS:1577:0:0:0:3:0
```

The value provided in the `ID` field reflects the SV association graph edge(s) from which the SV or indel was discovered. The value is unique within any single VCF output file produced by the SV Caller, and is used to link associated breakend records using the standard VCF `MATEID` key. You can use the entire value as a unique key, but parsing the key could lead to incompatibility with future DRAGEN versions. See the DRAGEN Bio-It Platform support site for information on the latest version of DRAGEN.

Convert SV VCF to BEDPE Format

It might be convenient to express structural variants in BEDPE format. For such applications, DRAGEN recommends the script `vcfToBedpe` available on GitHub. The repository is forked from @hall-lab with modifications to support VCF 4.2 SV format.

BEDPE format greatly reduces structural variant information compared to the SV Caller VCF output. In particular, breakend orientation, breakend homology, and insertion sequence are lost, in addition to the ability to define fields for locus and sample specific information. For this reason, Illumina only recommends BEDPE as a temporary output for applications that require it.

Statistics Output File

Additional statistics are provided in the files in `<output-directory>/sv/results/stats`.

File	Description
<code>alignmentStatsSummary.txt</code>	Fragment length quantiles for each input alignment file.
<code>svLocusGraphStats.tsv</code>	Statistics and runtime information pertaining to the SV locus graph.
<code>svCandidateGenerationStats.tsv</code>	Statistics and runtime information pertaining to the SV candidate generation.
<code>svCandidateGenerationStats.xml</code>	XML data backing the <code>svCandidateGenerationStats.tsv</code> report.
<code>diploidSV.sv_metrics.csv</code>	The number of passing SV calls under a diploid model. This file is only produced in the germline analysis, or tumor-normal analysis.
<code>somaticSV.sv_metrics.csv</code>	The number of passing SV calls under a somatic variant model. This file is only produced in the tumor-normal analysis.
<code>tumorSV.sv_metrics.csv</code>	The number of passing SV calls in the tumor-only analysis. This file is only produced in the tumor-only analysis.

Structural Variant De Novo Quality Scoring

You can enable *de novo* structural variant quality scoring in DRAGEN.

To enable *de novo* scoring for structural variant joint diploid calling, set `--sv-denovo-scoring` to true.

To adjust the threshold value for which variants are classified as *de novo*, use the `--sv-denovo-threshold` command line option. See [DN Field on page 318](#) for more information.

Inputs

De novo scoring requires the following two files:

- A pedigree file that specifies the relationship of all samples in the pedigree.
- The VCF output from germline structural variant calling analysis run jointly over all samples in the pedigree.

Pedigree File

The pedigree file is required for *de novo* scoring. Use the same file format as required for joint small variant calling analysis and *de novo* scoring. For information on the file format, see [Small Variant De Novo Calling on page 148](#). The file specifies which sample in the trio is the proband, mother, or father. If there are multiple trios specified in the pedigree file (eg, multigeneration pedigree or siblings), DRAGEN automatically detects the trios and provides the *de novo* scores on the proband sample of each detected trio.

Joint Germline Structural Variant VCF

DRAGEN applies *de novo* scoring to the VCF output from germline structural variant analysis for all samples specified in the pedigree file. You can supply the VCF file directly using the command line or produce the file as part of the DRAGEN run where *de novo* scoring is enabled.

Output

De novo scoring adds the *de novo* quality score (`DQ`) and *de novo* call (`DN`) fields for each sample in the output VCF file.

DQ Field

The `DQ` field is defined as follows.

```
##FORMAT=<ID=DQ,Number=1,Type=Float,Description="Denovo quality">
```

The `DQ` field represents the Phred-scaled posterior probability of the variant being *de novo* in the proband. For example, DQ scores of 13 and 20 correspond to a posterior probability of a *de novo* variant of 0.95 and 0.99. If DRAGEN can calculate the DQ score, the score is added to the proband samples. If the DQ score cannot be calculated, the field is set to " . ".

DN Field

The `DN` field is defined as follows.

```
##FORMAT=<ID=DN,Number=1,Type=String,Description="Possible values are 'DeNovo' or 'LowDQ'. Threshold for a passing de novo call is DQ >= 20">
```

DRAGEN compares valid (> 0) DQ scores against a threshold value. You can set the threshold value using the `--sv-denovo-threshold` command line option. For example, to set the threshold value to 10, add `--sv-denovo-threshold 10` to the command line. The default threshold value is 20.

If a DQ score is greater than or equal to the threshold value, the `DN` field is set to `DeNovo`. If the DQ score is below the threshold value, the `DN` field is set to `LowDQ`. If the DQ is 0 or `".`, the DQ score is invalid and the `DN` field is set to `".`.

De Novo Scoring Workflows

You can use *de novo* structural variant scoring in the following workflows.

- Perform *de novo* scoring in two DRAGEN runs. In the first, run germline structural variant analysis jointly over all samples in the pedigree file. In the second, apply *de novo* structural variant scoring to the joint germline VCF output. See [Two-Run Workflow on page 318](#).
- Perform *de novo* scoring in one DRAGEN run. Run germline structural variant analysis jointly over all samples in the pedigree file, and then apply *de novo* scoring to the joint germline structural variant calls. See [One-Run Workflow on page 319](#).

Two-Run Workflow

In the two-run workflow, first run a standard DRAGEN joint germline analysis over multiple samples as shown in the following example.

```
dragen -f \
--ref-dir <HASH_TABLE> \
--bam-input <BAM1> \
--bam-input <BAM2> \
--bam-input <BAM3> \
--enable-map align false \
--enable-sv true \
--output-directory <OUT_DIR1> \
--output-file-prefix <PREFIX1>
```

In the second run, use the VCF output (`<OUT_DIR1>/<PREFIX1>.sv.vcf.gz`) as input for *de novo* scoring. You can provide the VCF input using the `--variant` option. The following command line provides an example of the second run.

```
dragen -f \
```

```
--variant <MULTI_SAMPLE_VCF_FILE> \
--pedigree-file <PED_FILE> \
--enable-map-align false \
--sv-denovo-scoring true \
--output-directory <OUT_DIR2> \
--output-file-prefix <PREFIX2>
```

The resulting output VCF file (`<OUT_DIR2>/<PREFIX2>.sv.vcf.gz`) includes all *de novo* scoring annotations.

One-Run Workflow

Run a standard DRAGEN joint germline analysis over multiple samples with all required *de novo* scoring options. The following example shows the one-run workflow.

```
dragen -f \
--ref-dir <HASH_TABLE> \
--bam-input <BAM1> \
--bam-input <BAM2> \
--bam-input <BAM3> \
--enable-map align=false \
--enable-sv=true \
--output-directory <OUT_DIR> \
--output-file-prefix <PREFIX> \
--sv-denovo-scoring true \
--pedigree-file <PED_FILE>
```

The resulting output VCF file (`<OUT_DIR>/<PREFIX>.sv.vcf.gz`) includes all *de novo* scoring annotations.

Ploidy Estimator

The Ploidy Estimator runs by default. The Ploidy Estimator uses reads from the mapper/aligner to calculate the sequencing depth of coverage for each autosome and allosome in the human genome. The sex karyotype of the sample is then estimated using the ratios of the median sex chromosome coverages to the median autosomal coverage. The sex karyotype is estimated based on the range the ratios fall in. If the ratios are outside all expected ranges, then the Ploidy Estimator does not determine a sex karyotype.

Sex Karyotype	X Ratio Min	X Ratio Max	Y Ratio Min	Y Ratio Max
XX	0.75	1.25	0.00	0.25
XY	0.25	0.75	0.25	0.75
XXY	0.75	1.25	0.25	0.75
XYY	0.25	0.75	0.75	1.25
X0	0.25	0.75	0.00	0.25
XXXY	1.25	1.75	0.25	0.75
XXX	1.25	1.75	0.00	0.25

Ploidy estimation can fail if the type of input sequencing data cannot be determined to be either WGS or WES. When ploidy estimation fails the estimated median coverage values will be zero. The type of input sequencing data is determined using coverage skewness.

skewness = std::abs(autosomeMean - autosomeMedian) / autosomeMean

- If skewness <= 0.2 then the data is determined to be WGS.

 At least 2x coverage is required for WGS. If the coverage is less than 2x then the data may be treated improperly as WES.

- If skewness >=0.6 then the data is determined to be WES.
- If skewness > 0.2 and < 0.6 then it will have an undefined input sequencing data type, and the reported estimated median coverage values will be zero.

For WES data, the median exome coverage is estimated using the 99th percentile of coverage bins across each contig. This estimated median exome coverage is then reported by the Ploidy Estimator and used for sex estimation.

If there is not sufficient sequencing coverage in the autosomes (at least 2x for either WGS or WES) then the Ploidy Estimator does not determine a sex karyotype.

When both tumor and matched normal reads are provided as input, the Ploidy Estimator only estimates sequencing coverage and sex karyotype for the matched normal sample and ignores the tumor reads. If only tumor reads are provided as input, the Ploidy Estimator estimates sequencing coverage and sex karyotype for the tumor sample.

Output Metrics

The Ploidy Estimator results, including each normalized per-contig median coverage, is reported in the `<output-file-prefix>.ploidy_estimation_metrics.csv` file and in standard output.

The following is an example of the results.

```
PLOIDY ESTIMATION Autosomal median coverage 44.79
```

```
PLOIDY ESTIMATION X median coverage 42.47
PLOIDY ESTIMATION Y median coverage 20.82
PLOIDY ESTIMATION 1 median / Autosomal median 0.95
PLOIDY ESTIMATION 2 median / Autosomal median 1.05
PLOIDY ESTIMATION 3 median / Autosomal median 1.01
PLOIDY ESTIMATION 4 median / Autosomal median 0.99
...
PLOIDY ESTIMATION 22 median / Autosomal median 0.99
PLOIDY ESTIMATION X median / Autosomal median 0.95
PLOIDY ESTIMATION Y median / Autosomal median 0.46
PLOIDY ESTIMATION Ploidy estimation XXY
```

Ploidy Caller

The Ploidy Caller uses the per contig median coverage values from the Ploidy Estimator to detect aneuploidy and chromosomal mosaicism in a human germline sample from whole genome sequencing data.

The Ploidy Caller runs by default except in the following circumstances:

- The Ploidy Estimator cannot determine if the input data is from whole genome sequencing. For example, data from exome or targeted sequencing.
- The reference genome does not contain the expected 22 autosomes and two allosomes for humans.
- There is no germline sample. For example, tumor-only analysis.

Calling Model

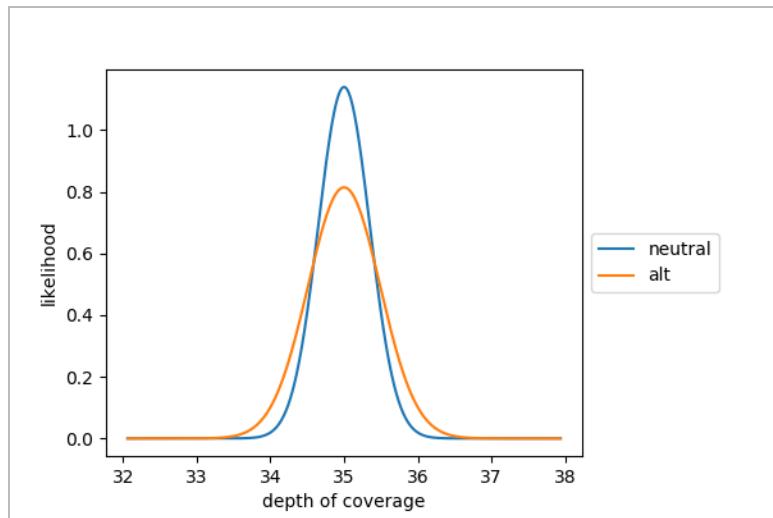
Chromosomal mosaicism is detected when there is a significant shift in median coverage of a chromosome compared to the overall autosomal median coverage.

The following table displays some examples of expected shifts in coverage for a given aneuploidy and mosaic fraction.

Neutral Copy Number	Variant Copy Number	Mosaic Fraction	Expected Coverage Shift
2	1	10 %	-5%
2	1	5 %	-2.5%
2	3	5 %	+2.5%
2	3	10 %	+5%

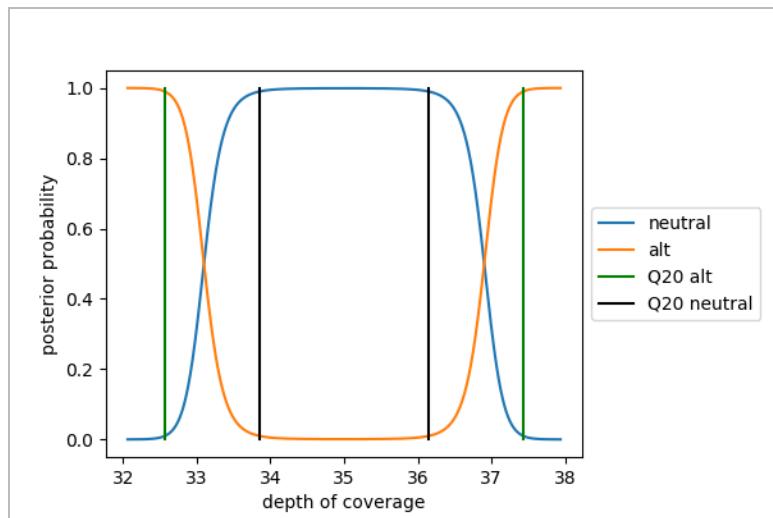
The Ploidy Caller models coverage as a normal distribution for both the null (neutral) and the alternative (mosaic) hypotheses. The two normal distributions have equal mean at the median autosomal coverage for the sample, but the variance of the alternative normal distribution is greater than that of the null normal distribution. The baseline variance of the two models at 30x coverage was determined empirically from a cohort of ~2500 WGS samples. The actual variance used for the two models is calculated from the baseline variance at 30x coverage, adjusting for the median autosomal coverage of the sample. Below are the likelihood distributions for the null and alternative hypotheses for a sample with 35x median autosomal coverage.

Figure 11 Null and Alternative Hypotheses Likelihood Distributions



After applying an empirically estimated prior for chromosomal mosaicism, the Ploidy Caller generates ploidy calls according to the posterior probability of null and alternative hypotheses as shown below for a sample with 35x median autosomal sequencing coverage.

Figure 12 Null and Alternative Hypotheses Posterior Probability



A 35x median autosomal coverage, the threshold for deciding between a neutral (REF) and an alternative (DEL or DUP) call is roughly at +/- 5% shift in coverage for an autosome. A 100x median autosomal coverage, the threshold is at roughly +/- 3% shift in coverage for an autosome. A Q20 threshold is used to filter low quality calls.

Ploidy Caller Reference Sex Karyotype

In addition to detecting aneuploidy and chromosomal mosaicism in autosomes where the expected reference ploidy is two, the Ploidy Caller can also detect these variants in allosomes.

The reference sex karyotype used for making calls on the allosomes is determined from the sex karyotype of the sample either provided on the command line using `--sample-sex` or from the Ploidy Estimator. If the sex karyotype of the sample is not provided on the command line and not determined by the Ploidy Estimator, then the sex karyotype is assumed to be XX. Whenever the sex karyotype contains at least one Y chromosome, the reference sex karyotype is XY. If the sex karyotype does not contain at least one Y chromosome, then the sex karyotype is XX.

The following table displays each of the possible sex karyotypes for a sample. If the Y chromosome reference ploidy is zero, then ploidy calling is not performed on the Y chromosome.

Sex Karyotype	X Reference Ploidy	Y Reference Ploidy
XX	2	0
XY	1	1
XXY	1	1
XYY	1	1
X0	2	0
XXXY	1	1
XXX	2	0

Ploidy Caller Output File

The Ploidy Caller generates a <output-file-prefix>.ploidy.vcf.gz output file in the output directory. The output file follows the VCF 4.2 Specification. A single record is reported for each reference autosome and allosome, except for the Y chromosome if the reference sex karyotype is XX. Calls are not made for other sequences in the reference genome, such as mitochondrial DNA, unlocalized or unplaced sequences, alternate contigs, decoy contigs, or the Epstein-Barr virus sequence. The VCF header is annotated with `##source=DRAGEN_PLOIDY` to indicate the file is generated by the DRAGEN PLOIDY pipeline.

The following information is provided in the VCF file.

- Meta information—The VCF output file contains common meta-information such as DRAGENVersion and DRAGEN CommandLine, as well as Ploidy Caller specific information. The VCF header contains the meta-information for median autosome depth of coverage, the provided sex karyotype if available, the estimated sex karyotype from the Ploidy Estimator if available, and the reference sex karyotype. The following is an example of the header lines:

```
##autosomeDepthOfCoverage=36.635
##providedSexKaryotype=XY
##estimatedSexKaryotype=X0
##referenceSexKaryotype=XY
```

- FILTER fields—The VCF output file includes the LowQual filter, which filters results with quality score below 20.
- INFO Fields—The VCF output INFO fields include the following:
 - END—End position of the variant described in this record.
 - SVTYPE—Type of structural variant.
- Format fields—The VCF output file includes the following format fields. There is no GT FORMAT field. A variant call in the VCF displays either <DUP> or in the ALT column. A non-variant call displays . in the ALT column. If using the output file for downstream use, a GT field can be added for variant calls using ./1 for a diploid contig and 1 for a haploid contig. For non-variant calls, use 0/0 for diploid and 0 for haploid.
 - DC—Depth of coverage.
 - NDC—Normalized depth of coverage.

The following is an example output file.

```
##fileformat=VCFv4.2
...
##autosomeDepthOfCoverage=36.635
##providedSexKaryotype=XY
##estimatedSexKaryotype=X0
##referenceSexKaryotype=XY
```

```

##INFO=<ID=END,Number=1,Type=Integer,Description="End position of the variant described in
this record">

##INFO=<ID=SVTYPE,Number=1,Type=String,Description="Type of structural variant">

##ALT=<ID=DEL,Description="Deletion relative to the reference">

##ALT=<ID=DUP,Description="Region of elevated copy number relative to the reference">

##FILTER=<ID=LowQual,Description="QUAL below 20">

##FORMAT=<ID=DC,Number=1,Type=Float,Description="Depth of coverage">

##FORMAT=<ID=NDC,Number=1,Type=Float,Description="Normalized depth of coverage">

#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT MySampleName
chr1 1 . N . 31.1252 PASS END=248956422 DC:NDC 36.836:1.00549
chr2 1 . N . 31.451 PASS END=242193529 DC:NDC 36.668:1.0009
...
chr21 1 . N . 31.4499 PASS END=46709983 DC:NDC 36.6:0.999045
chr22 1 . N . 28.8148 PASS END=50818468 DC:NDC 37.2:1.01542
chrX 1 . N . 29.7892 PASS END=156040895 DC:NDC 18:0.982667
chrY 1 . N <DEL> 150 PASS END=57227415;SVTYPE=DEL DC:NDC 5.7:0.31
1178

```

Cell Line Artifacts

Samples derived from cell lines frequently have coverage artifacts that might result in variant ploidy calls on some chromosomes. Chromosomes 17, 19, and 22 are the most common for the cell line coverage artifacts. When performing accuracy assessments of ploidy calls on cell line samples, filter out chromosomes with known cell line artifacts.

QC Metrics and Coverage/Callability Reports

DRAGEN generates pipeline-specific metrics coverage reports during each run. There are four different groups of metrics that are generated at different stages of the pipeline:

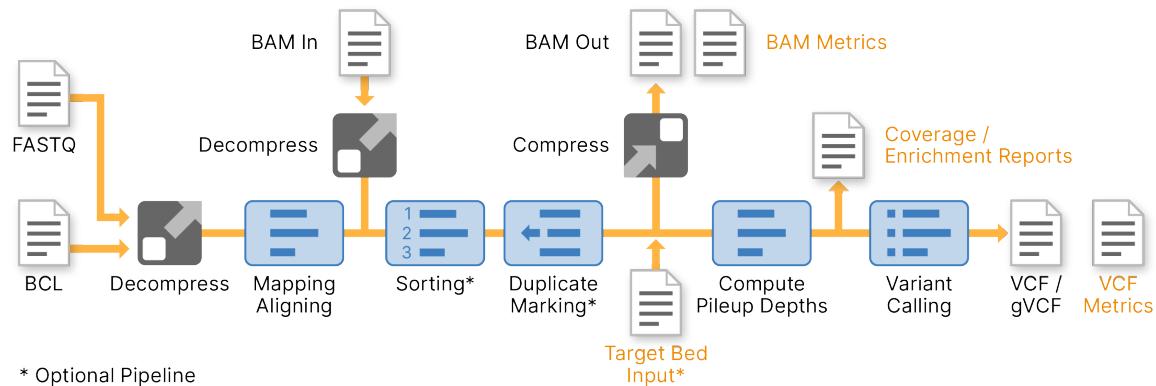
- Mapping and Aligning metrics
- VCF metrics

- Duration (or run time) metrics
- Coverage (or enrichment) metrics and reports

The mapping/aligning metrics, VCF metrics, Duration metrics, and a subset of available coverage reports are autogenerated and do not require any activation or specific commands. Additional coverage metrics can be enabled, and additional coverage regions can be specified.

DRAGEN performs metric calculation during analysis so that it does not impact the run time.

Figure 13 Generation of Metrics and Reports



QC Metrics Output Format

The QC metrics are printed to the standard output and CSV files are written to the run output directory.

- <output prefix>.mapping_metrics.csv
- <output prefix>.vc_metrics.csv
- <output prefix>.<coverage region prefix>_coverage_metrics.csv
- <output prefix>.time_metrics.csv
- <output prefix>.<other coverage reports>.csv

Section	RG/Sample	Metric	Count/Ration/Time	Percentage/Seconds
MAPPING/ALIGNING SUMMARY		Total input reads	816360354	
MAPPING/ALIGNING SUMMARY		Number of duplicate reads (marked not removed)	15779031	1.93

Section	RG/Sample	Metric	Count/Ration/Time	Percentage/Seconds
...				
MAPPING/ALIGNING PER RG	RGID_1	Total reads in RG	816360354	100
MAPPING/ALIGNING PER RG	RGID_1	Number of duplicate reads (marked)	15779031	1.93
...				
VARIANT CALLER SUMMARY		Number of samples	1	
VARIANT CALLER SUMMARY		Reads Processed	738031938	
...				
VARIANT CALLER PREFILTER	SAMPLE_1	Total	4918287	100
VARIANT CALLER PREFILTER	SAMPLE_1	Biallelic	4856654	98.75
...				
COVERAGE SUMMARY		Aligned bases in genome	14847587	100.00
COVERAGE SUMMARY		Average alignment coverage over genome	0.03	
...				
RUN TIME		Time loading reference	00:18.6	18.65
RUN TIME		Time aligning reads	19:24.4	1164.42

Mapping and Aligning Metrics

Mapping and aligning metrics, such as the metrics computed by the Samtools Flagstat command, are available on an aggregate level (over all input data), and on a per read group level. Unless explicitly stated, the metrics units are in reads (ie, not in terms of pairs or alignments).

Metric	Description
Total input reads	Total number of reads in the input FASTQ files.
Number of duplicate marked reads	Reads marked as duplicates as a result of the --enable-duplicate-marking option being set to true.
Number of duplicate marked and mate reads removed	Reads marked as duplicates, along with any mate reads that are removed when the --remove-duplicates option is set to true.
Number of unique reads	Total number of reads minus the duplicate marked reads.
Reads with mate sequenced	Number of reads with a mate.
Reads without mate sequenced	Total number of reads minus number of reads with mate sequenced.
QC-failed reads	Reads that did not pass platform/ vendor quality checks (SAM flag 0x200).
Mapped reads	Total number of mapped reads minus number of unmapped reads.
Number of unique and mapped reads	Number of mapped reads minus number of duplicate marked reads.
Unmapped reads	Total number of reads that could not be mapped.
Singleton reads	Number of reads where the read could be mapped, but the paired mate could not be read.
Paired reads	Count of reads in which both reads in the pair are mapped.
Properly paired reads	Both reads in the pair are mapped and fall within an acceptable range from each other based on the estimated insert length distribution.
Not properly paired reads (discordant)	The number of paired reads minus the number of properly paired reads.
Paired reads mapped to different chromosomes	The number of reads with a mate, where the mate was mapped to a different chromosome.

Metric	Description
Paired reads mapped to different chromosomes (MAPQ >= 10)	The number of reads with a MAPQ > 10 and with a mate, where the mate was mapped to a different chromosome.
Reads with indel R1	The percentage of R1 reads containing at least 1 indel.
Reads with indel R2	The percentage of R2 reads containing at least 1 indel.
Soft-clipped bases R1	The percentage of bases in R1 reads that are soft-clipped.
Soft-clipped bases R2	The percentage of bases in R2 reads that are soft-clipped.
Mismatched bases R1	The number of mismatched bases on R1, which is the sum of SNP count and indel lengths. The metric does not count anything within soft clipping or RNA introns. The metric also does not count a mismatch if either the reference base or read base is N.
Mismatched bases R2	The number of mismatched bases on R2, which is the sum of SNP count and indel lengths. The metric does not count anything within soft clipping or RNA introns. The metric also does not count a mismatch if either the reference base or read base is N.
Mismatched bases R1 (excluding indels)	The number of mismatched bases on R1. The indels lengths are ignored. It does not count anything within soft clipping or RNA introns. The metric also does not count a mismatch if either the reference base or read base is N.
Mismatched bases R2 (excluding indels)	The number of mismatched bases on R2. The indels lengths are ignored. The metric does not count anything within soft clipping or RNA introns. The metric also does not count a mismatch if either the reference base or read base is N.
Q30 Bases	The total number of bases with a BQ ≥ 30 .
Q30 Bases R1	The total number of bases on R1 with a BQ ≥ 30 .
Q30 Bases R2	The total number of bases on R2 with a BQ ≥ 30 .
Q30 Bases (excluding dups and clipped bases)	The number of bases on nonduplicate and nonclipped bases with a BQ ≥ 30 .
Histogram of reads map qualities	<ul style="list-style-type: none"> • Reads with MAPQ [40:inf) • Reads with MAPQ [30:40) • Reads with MAPQ [20:30) • Reads with MAPQ [10:20) • Reads with MAPQ [0:10)
Total alignments	Total number of loci reads aligned to with > 0 quality.

Metric	Description
Secondary alignments	Number of secondary alignment loci.
Supplementary (chimeric) alignments	A chimeric read is split over multiple loci (possibly due to structural variants). One alignment is referred to as the representative alignment. The other are supplementary.
Estimated read length	Total number of input bases divided by the number of reads.
Histogram	See Histogram Coverage Report on page 340 .
PCT of bases aligned that fell inside the interval region	Number of bases inside the interval region and the target region divided by the total number of bases aligned.
Estimated sample contamination	The estimated fraction of reads in a sample that may be from another human source.

The prediction accuracy of variant calling is affected by cross-sample contamination. Even small levels of contamination can lead to many FP calls, especially in pipelines where the aim is to detect variants with low allele frequencies.

The DRAGEN cross-sample contamination module uses a probabilistic mixture model to estimate the fraction of reads in a sample that may be from another human source. This sample contamination fraction is estimated as the parameter value in the mixture model that maximizes the likelihood of the observed reads at multiple pile up locations. The mixture model accounts for the population allele frequencies and the inferred sample genotypes.

To enable this metric in germline mode, you must provide the file path on the command line to a VCF that includes marker sites (RSIDs) with population allele frequencies.

```
--qc-cross-cont-vcf /opt/edico/config/sample_cross_contamination_resource_[hg19 or GRCh37 or GRCh38].vcf
```

In somatic mode the contamination algorithm first tries to avoid biases that could be introduced by CNV or LoH. The algorithm also estimates nucleotide noise from the sample to adjust for FFPE samples.

To enable somatic contamination detection, use the following setting.

```
--qc-somatic-contam-vcf /opt/edico/config/somatic_sample_cross_contamination_resource_[hg19 or GRCh37 or GRCh38].vcf.gz
```

The VCF resource files that are included with DRAGEN can be reconstructed from the Ensembl database. The VCF files included in the DRAGEN config folder contain ~5000 marker locations where the population AFs are close to 0.5. The files are reference-specific (hg19/GRCh37/hg38). DRAGEN will abort if an incompatible resource and reference file is used (eg, GRCh37 resource file and hg19 reference).

The following shows example output for a sample with 1.1% estimated contamination. This value is provided as a fraction, so a value of 0.011 is the same as 1.1%.

```
MAPPING/ALIGNING SUMMARY Estimated sample contamination 0.011
```

The precision of variant calling, particularly for somatic variants, can be significantly impacted by cross-sample contamination. To ensure safe usage of a sample, the level of cross-sample contamination must be considerably lower than the minimum allele frequencies of interest. For instance, if a sample has 1% contamination, it may be necessary to disregard all variants with less than 5% Fallele frequency. The cross-contamination metric for a sample reaches saturation near 30% contamination.

The contamination module requires a minimum of 100 valid pileups for contamination estimation, where a pileup is considered valid if it has at least 10X coverage and 95% or more reads are deemed valid. Soft clipped reads that could indicate INDELS or structural variants are not considered valid, and datasets with untrimmed adapters may lead to most reads being soft clipped and classified as invalid. If the contamination module reports "NA," even for high-coverage samples, it is recommended to inspect a few pileup locations in IGV for evidence of untrimmed bases.

Other settings to consider include:

`qc-contam-min-cov`: the minimum read coverage required for a pileup to be used in contamination detection (default is 10). Coverage below 5-10X may produce unreliable results. `qc-contam-min-valid-read-ratio`: the minimum ratio of valid reads in a pileup for it to be considered valid. The default setting is 0.95, meaning 95% of the reads in a pileup must be valid. This value may be lowered to 0.75 and still yield accurate contamination estimates. If many reads are classified as invalid, it is likely due to untrimmed adapters that are being systematically soft clipped. It is recommended to fix the BAM file rather than force the contamination module to use these reads.

Somatic Mode Mapping and Aligning Metrics

In somatic tumor-normal mode, the mapping and aligning metrics are generated separately for the tumor and normal samples, with each line beginning with TUMOR or NORMAL to indicate the sample. The metrics for the tumor sample are output first, followed by the metrics for the normal sample. Metrics per read group are also separated into tumor and normal read groups.

In somatic tumor-only mode, the mapping and aligning metrics follow the same conventions as germline mode, without metrics labeled as TUMOR or NORMAL.

Variant Calling Metrics

The generated variant calling metrics are similar to the metrics computed by RTG vcfstats. Metrics are reported for each sample in multi sample VCF and gVCF files. Based on the run case, metrics are reported either as standard VARIANT CALLER or JOINT CALLER. Metrics are reported both for the raw (PREFILTER) and hard filtered (POSTFILTER) VCF file.

Panel of Normals (PON) and COSMIC filtered variants are counted as PASS variants in the POSTFILTER VCF metrics. These PASS variants can cause higher than expected variant counts in the POSTFILTER VCF metrics.

Metric	Description
Number of samples	Number of samples in the population/ joint VCF.
Reads Processed	The number of reads used for variant calling, excluding any duplicate marked reads and reads falling outside of the target region.
Total	The total number of variants (SNPs + MNPs + indels).
Biallelic	Number of sites in a genome that contains two observed alleles. The reference is counted as one allele, which allows for one variant allele.
Multiallelic	Number of sites in the VCF that contain three or more observed alleles. The reference is counted as one, which allows for two or more variant alleles.
SNPs	A variant is counted as an SNP when the reference, allele 1, and allele 2 are all length 1.
Insertions (Hom)	Number of variants that contains homozygous insertions.
Insertions (Het)	Number of variants where both alleles are insertions, but not homozygous.
Deletions (Het)	Number of variants that contains homozygous deletions.
INDELS (Het)	Number of variants where genotypes are either [insertion+deletion], [insertion+SNP], or [deletion+SNP].
De Novo SNPs	<i>De novo</i> marked SNPs with DQ > 0.05. Set the <code>--qc-snp-denovo-quality-threshold</code> option to the required threshold. The default is 0.05.
De Novo INDELS	<i>De novo</i> marked indels with DQ values > 0.02. This DQ threshold can be specified by setting the <code>--qc-indel-denovo-quality-threshold</code> option to the required DQ threshold. The default is 0.02.

Metric	Description
De Novo MNPs	<i>De novo</i> marked SNPs with DQ > 0.05. Set the <code>--qc-snp-denovo-quality-threshold</code> to the required threshold. The default is 0.05.
(Chr X SNPs)/(Chr Y SNPs) ratio in the genome (or the target region)	Number of SNPs in chromosome X (or in the intersection of chromosome X with the target region) divided by the number of SNPs in chromosome Y (or in the intersection of chromosome Y with the target region). If there was no alignment to either chromosome X or chromosome Y, this metric shows as NA.
SNP Transitions	An interchange of two purines (A<->G) or two pyrimidines (C<->T).
SNP Transversions	An interchange of purine and pyrimidine bases Ti/Tv ratio: ratio of transitions to transitions.
Heterozygous	Number of heterozygous variants.
Homozygous	Number of homozygous variants.
Het/Hom ratio	Heterozygous/ homozygous ratio.
In dbSNP	Number of variants detected that are present in the dbSNP reference file. If no dbSNP file is provided via the <code>--bsnp</code> option, then both the In dbSNP and Novel metrics show as NA.
Novel	Total number of variants minus number of variants in dbSNP.
Percent Callability	Available in germline and somatic modes with gVCF output. The percentage of non-N reference positions having a PASSing genotype call. Multiallelic variants are not counted. Deletions are counted for all the deleted reference positions only for homozygous calls. Only autosomes and chromosomes X, Y, and M are considered. To produce this metric for non-human references, set <code>--qc-callability-autosome-contigs</code> to specify the autosome contig names. Optionally, <code>--qc-callability-xym-contigs</code> allows setting X, Y and M contig names.
Percent Autosome Callability	Only autosomes are considered. To produce this metric for non-human references, set <code>--qc-callability-autosome-contigs</code> to specify the autosome contig names.
Percent QC Region Callability in Region i (i is equivalent to regions 1, 2, or 3)	Available if callability for custom regions is requested via the <code>--qc-coverage-region-i</code> option and the callability output is specified with <code>--qc-coverage-reports-i</code> . All contigs are considered. Setting <code>--qc-callability-autosome-contigs</code> enables outputting this metric for non-human references.

Per Contig Het/Hom Ratio

When the germline small variant caller is executed, DRAGEN calculates a per het/hom ratio per contig. DRAGEN reports the ratios for both the raw (PREFILTER) and hard-filtered (POSTFILTER) VCF. The metrics are output to the `.vc_hethom_ratio_metrics.csv` file. The file contains the following values for each primary contig processed.

- Contig
- Number of heterozygous variants
- Number of homozygous variants
- Het/Hom ratio

The following example shows a section of the metrics.

```
VARIANT CALLER POSTFILTER,HG04070,1 Heterozygous,185733
VARIANT CALLER POSTFILTER,HG04070,1 Homozygous,182928
VARIANT CALLER POSTFILTER,HG04070,1 Het/Hom ratio,1.015
VARIANT CALLER POSTFILTER,HG04070,2 Heterozygous,203946
VARIANT CALLER POSTFILTER,HG04070,2 Homozygous,174294
VARIANT CALLER POSTFILTER,HG04070,2 Het/Hom ratio,1.170
VARIANT CALLER POSTFILTER,HG04070,3 Heterozygous,192861
VARIANT CALLER POSTFILTER,HG04070,3 Homozygous,130087
VARIANT CALLER POSTFILTER,HG04070,3 Het/Hom ratio,1.483
VARIANT CALLER POSTFILTER,HG04070,4 Heterozygous,178389
VARIANT CALLER POSTFILTER,HG04070,4 Homozygous,157062
VARIANT CALLER POSTFILTER,HG04070,4 Het/Hom ratio,1.136
```

You can use the het/hom ratio values as an indication of whole chromosome uniparental disomy (UPD). UPD of certain chromosomes are associated with genetic syndromes known as imprinting disorders. Whole chromosome UPD have het/hom ratios close to 0.0. Ranges vary, but are usually between 1.0–2.0. Make sure you are interpreting het/hom ratios in the context of your own sequencing data.

Duration Metrics

The duration metrics section includes a breakdown of the run duration for each process. For example, the following metrics are generated for the mapper and variant caller pipeline:

- Time loading reference
- Time aligning reads
- Time sorting and marking duplicates
- Time DRAGStr calibration
- Time partial reconfiguration

- Time variant calling
- Total run time

Callability Report

DRAGEN automatically generates a callability report as part of variant caller metrics when running the small variant caller with gVCF output mode. DRAGEN appends the percent callability for each region included in the BED file specified by the test and generates a full-callability-report in both BED and CSV formats. Callability is defined as the fraction of non-N reference positions having a PASSing genotype call. The following criteria are used to calculate callability metrics:

- Callability is calculated over the gVCF.
- Multi-allelic variants are not counted.
- Decoy contigs are ignored.
- Unplaced and unlocalized contigs are ignored.
- N positions are considered non-callable.
- For regions where no variant calling was performed, callability is 0.
- A homozygous deletion counts as a PASSing genotype call for all the reference positions spanned by the deletion.

If the `--vc-target-bed` option is specified, the output is a `target_bed_callability.bed` file that contains the overall and autosome callability over the input target bed region. The padding size specified by the `--vc-target-bed-padding` option is used and overlapping regions are merged.

Callability can also be output with the custom region coverage/callability reports.

Coverage/Callability Reports Over Custom Regions

DRAGEN generates the following coverage reports:

- A set of default reports for either the whole genome or the target region, if the `--vc-target-bed` option is specified.
- Optionally, additional reports for up to three regions of interest (coverage regions).

For each specified region, DRAGEN generates the default reports and any optional report requested for the region.

To generate coverage region reports, use the `-qc-coverage-region-i` option, where i is 1, 2, or 3.

- Each `-qc-coverage-region-i` option requires a BED file argument.
- Regions in each BED file can be optionally padded using the `--qc-coverage-region-padding-i` option, which defaults to 0.
- A set of default reports is generated for each region.
- Optionally, additional reports can be specified for each region by using the `-qc-coverage-reports-i` option.

The following example shows the options required to generate coverage reports.

```
$ dragen ... \
--qc-coverage-region-1 <bed file 1> \
--qc-coverage-reports-1 full_res \
--qc-coverage-region-2 <bed file 2> \
--qc-coverage-region-3 <bed file 3> \
--qc-coverage-reports-3 full_res cov_report
```

Counting Reads and Bases

All default and optional coverage reports listed in [Table 8](#) and [Table 9](#) use the following default rules for counting reads and bases:

- Duplicate reads are ignored.
- Soft and/or hard-clipped bases are ignored.
- Overlapping mates are double-counted.
- Reads with MAPQ > 0 are included. MAPQ=0 reads are filtered.
- BQ ≥ 0 are included.

Nondefault settings:

The reports are available with or without running either the mapper and aligner, or the variant caller. However, the `--enable-sort` options must be set to true (the default is true).

By default overlapping mates are double counted. Set `--qc-coverage-ignore-overlaps=true` to resolve all of the alignments for each fragment and avoid double-counting any overlapping bases. This might result in marginally longer run times. This option also requires setting `--enable-map-align=true`. `--qc-coverage-ignore-overlaps` is a global setting and updates all qc-coverage-reports.

By default soft-clipped bases are not counted towards coverage. Set `--qc-coverage-count-soft-clipped-bases=true` to include soft-clipped bases in the coverage calculations. `--qc-coverage-count-soft-clipped-bases` is a global setting and updates all qc-coverage reports.

Any combination of the optional reports can be requested for each region. If multiple report types are selected per region, they should be space-separated.

It is possible to override the min MAPQ and min BQ to apply for a given region using the qc-coverage-filters.

A coverage filter is enabled by using one of the `--qc-coverage-filters-i` options (where i is 1, 2, or 3), in combination with the associated `--qc-coverage-region-i` option. The default value for `--qc-coverage-filters-i` is `mapq<1, bq<0`. The default includes all BQ, but filters reads with MAPQ=0.

- `--qc-coverage-region-i=<targetedregions.bed>`
- `--qc-coverage-filters-i <filters string>`

For example, the following options are used to enable 1 bp resolution coverage output with filtering:

```
--qc-coverage-region-1 <targetedregions.bed>
```

```
--qc-coverage-filters-1 'mapq<10,bq<30'
--qc-coverage-reports-1 full_res
```

- The argument syntax is mapq<value,bq<value, which means that reads that have a mapping quality less than the specified value are not counted, and/or bases with a base call quality below the specified value.
- Valid filter arguments are mapq and bq only. Either, or both, can be specified.
- Only one operator < is supported. <=, >, >=, = are not supported.
- When filtering is enabled for a targeted region, DRAGEN outputs the filtered report files for this region. Unfiltered report files are not output for the targeted filtered region.

Callability for the Custom Regions

If the `--qc-coverage-region-i` option is used with `--qc-coverage-reports-i` (where i is 1, 2, or 3), callability can be added as a report type for that region. The output is a `qc-coverage-region-i_callability.bed` file. For each specified `qc-coverage-region-i` file, the average callability is reported in the variant calling metrics file. The padding size specified by the `--qc-coverage-region-padding-i` is used and overlapping regions are merged.

The optional min MAPQ and min BQ filters only influence read and base counting and do not influence the callability reports.

Contig lengths and region of interest lengths (used as denominators) do not include regions with N in the FASTA.

Available Report Types

Table 8 Default Report

File Name	Description
<code>_coverage_metrics.csv</code>	<i>Coverage Metrics Report</i> on page 338
<code>_fine_hist.csv</code>	<i>Fine Histogram Coverage Report</i> on page 340
<code>_hist.csv</code>	<i>Histogram Coverage Report</i> on page 340
<code>_overall_mean_cov.csv</code>	<i>Overall Mean Coverage Report</i> on page 341
<code>_contig_mean_cov.csv</code>	<i>Per Contig Mean Coverage Report</i> on page 341
<code>_ploidy.csv</code>	<i>Predicted Ploidy Report</i>

File Name	Description
_read_cov_report.bed	Read Coverage Report on page 346

Table 9 Optional Reports

File Name	Description
_full_res.bed	Full Res Report on page 341
_cov_report.bed	Coverage Report on page 342
_callability.bed	Callability Report on page 335

Example to enable all three the optional reports over the targeted region:

```
--qc-coverage-region-1 <targetedregions.bed>
--qc-coverage-reports-1 full_res cov_report callability
```

Coverage Metrics Report

The coverage metrics report outputs a `_coverage_metrics.csv` file, which provides metrics over a region, where the region can be the genome, a target region, or a QC coverage region. The first column of the output file contains the section name COVERAGE SUMMARY and the second column is empty for all metrics.

The following criteria are used when calculating coverage:

- Duplicate reads and clipped bases are ignored.
- By default, reads with MAPQ < 1 and bases with BQ < 0 are ignored. You can use the `qc-coverage-filters-n` option to specify which BQ bases and MAPQ reads to filter out.

The following table lists the calculated metrics:

Metric	Description
Aligned bases in region	Number of uniquely mapped bases to region and the percentage relative to the number of uniquely mapped bases to the genome.
Average alignment coverage over region	Number of uniquely mapped bases to region divided by the number of sites in region.
Uniformity of coverage (PCT > 0.2 * mean) over region	Percentage of sites with coverage greater than 20% of the mean coverage in region.
PCT of region with coverage [ix, inf)	Percentage of sites in region with at least ix coverage, where i can equal 100, 50, 20, 15, 10, 3, 1, or 0.

Metric	Description
PCT of region with coverage [ix, jx]	Percentage of sites in region with at least ix but less than jx coverage, where (i, j) can equal (50, 100), (20, 50), (15, 20), (10, 15), (3, 10), (1, 3), or (0, 1).
Average chromosome X coverage over region	Total number of bases that aligned to the intersection of chromosome X with region divided by the total number of loci in the intersection of chromosome X with region. If there is no chromosome X in the reference genome or the region does not intersect chromosome X, this metric shows as NA.
Average chromosome Y coverage over region	Total number of bases that aligned to the intersection of chromosome Y with region divided by the total number of loci in the intersection of chromosome Y with region. If there is no chromosome Y in the reference genome or the region does not intersect chromosome Y, this metric shows as NA.
XAvgCov/YAvgCov ratio over genome/target region	Average chromosome X alignment coverage in region divided by the average chromosome Y alignment coverage in region. If there is no chromosome X or chromosome Y in the reference genome or the region does not intersect chromosome X or Y, this metric shows as NA.
Average mitochondrial coverage over region	Total number of bases that aligned to the intersection of the mitochondrial chromosome with region divided by the total number of loci in the intersection of the mitochondrial chromosome with region. If there is no mitochondrial chromosome in the reference genome or the region does not intersect mitochondrial chromosome, this metric shows as NA.
Average autosomal coverage over region	Total number of bases that aligned to the autosomal loci in region divided by the total number of loci in the autosomal loci in region. If there is no autosome in the reference genome, or the region does not intersect autosomes, this metric shows as NA.
Median autosomal coverage over region	Median alignment coverage over the autosomal loci in region. If there is no autosome in the reference genome or the region does not intersect autosomes, this metric shows as NA.
Mean/Median autosomal coverage ratio over region	Mean autosomal coverage in region divided by the median autosomal coverage in region. If there is no autosome in the reference genome or the region does not intersect autosomes, this metric shows as NA.
Aligned reads in region	Number of uniquely mapped reads to region and percentage relative to the number of uniquely mapped reads to the genome. Only reads with MAPQ ≥ 1 are included. Secondary and supplementary alignments are ignored.

The following is an example of the contents of the `_coverage_metrics.csv` file:

```
COVERAGE SUMMARY,,Aligned bases,148169295474
COVERAGE SUMMARY,,Aligned bases in genome,148169295474,100.00
COVERAGE SUMMARY,,Average alignment coverage over genome,46.08
COVERAGE SUMMARY,,Uniformity of coverage (PCT > 0.2*mean) over genome,91.01
COVERAGE SUMMARY,,PCT of genome with coverage [100x: inf),0.25
COVERAGE SUMMARY,,PCT of genome with coverage [ 50x: inf),50.01
COVERAGE SUMMARY,,PCT of genome with coverage [ 20x: inf),89.46
COVERAGE SUMMARY,,PCT of genome with coverage [ 15x: inf),90.51
COVERAGE SUMMARY,,PCT of genome with coverage [ 10x: inf),91.01
COVERAGE SUMMARY,,PCT of genome with coverage [ 3x: inf),91.69
COVERAGE SUMMARY,,PCT of genome with coverage [ 1x: inf),92.10
COVERAGE SUMMARY,,PCT of genome with coverage [ 0x: inf),100.00
COVERAGE SUMMARY,,PCT of genome with coverage [ 50x:100x),49.76
COVERAGE SUMMARY,,PCT of genome with coverage [ 20x: 50x),39.45
COVERAGE SUMMARY,,PCT of genome with coverage [ 15x: 20x),1.04
COVERAGE SUMMARY,,PCT of genome with coverage [ 10x: 15x),0.51
COVERAGE SUMMARY,,PCT of genome with coverage [ 3x: 10x),0.67
COVERAGE SUMMARY,,PCT of genome with coverage [ 1x: 3x),0.42
COVERAGE SUMMARY,,PCT of genome with coverage [ 0x: 1x),7.90
COVERAGE SUMMARY,,Average chr X coverage over genome,24.70
COVERAGE SUMMARY,,Average chr Y coverage over genome,20.96
COVERAGE SUMMARY,,Average mitochondrial coverage over genome,20682.19
COVERAGE SUMMARY,,Average autosomal coverage over genome,47.81
COVERAGE SUMMARY,,Median autosomal coverage over genome,48.62
COVERAGE SUMMARY,,Mean/Median autosomal coverage ratio over genome,0.98
COVERAGE SUMMARY,,XAvgCov/YAvgCov ratio over genome,1.18
COVERAGE SUMMARY,,XAvgCov/AutosomalAvgCov ratio over genome,0.52
COVERAGE SUMMARY,,YAvgCov/AutosomalAvgCov ratio over genome,0.44
COVERAGE SUMMARY,,Aligned reads,1477121058
COVERAGE SUMMARY,,Aligned reads in genome,1477121058,100.00
```

Fine Histogram Coverage Report

The fine histogram report outputs a `_fine_hist.csv` file, which contains two columns: Depth and Overall. The value in the Depth column ranges from 0 to 1000+ and the Overall column indicates the number of loci covered at the corresponding depth.

Masked regions in the FASTA are ignored and no depth for these regions are reported.

Histogram Coverage Report

The histogram report outputs a `_hist.csv` file, which provides the following information:

- Percentage of bases in the coverage BED/target BED/WGS region that fall within a certain range of coverage.
- Duplicate reads are ignored if DRAGEN is run with `--enable-duplicate-marking true`.

The following ranges are used:

```
"[100x:inf)", "[1x:3x)", "[0x:1x]"
```

Masked regions in the FASTA are ignored and no depth for these regions are reported.

Overall Mean Coverage Report

The Overall Mean Coverage report generates an `_overall_mean_cov.csv` file, which contains the average alignment coverage over the coverage BED/target BED/WGS, as applicable.

The following is an example of the contents of the `_overall_mean_cov.csv` file:

```
Average alignment coverage over target_bed, 80.69
```

Masked regions in the FASTA are ignored and no depth for these regions are reported.

Per Contig Mean Coverage Report

The Contig Mean Coverage report generates a `_contig_mean_cov.csv` file, which contains the estimated coverage for all contigs and an autosomal estimated coverage. The file includes the following three columns:

Column 1	Column 2	Column 3
Contig name	Number of bases aligned to that contig, which excludes bases from duplicate marked reads, reads with MAPQ=0, and clipped bases.	The estimated covered, calculated as follows. Number of bases aligned to the contig (ie, Col2) divided by length of the contig or (if a target BED is used) the total length of the target region spanning that contig.

Masked regions in the FASTA are ignored and no depth for these regions are reported.

Full Res Report

The Full Res Report outputs a `_full_res.bed` file in tab-delimited format. The first three columns are the standard BED fields, and the fourth column is the depth. Each record in the file is for a given interval that has a constant depth. If the depth changes, then a new record is written to the file. Alignments that have a mapping quality value of 0, duplicate reads, and clipped bases are not counted towards the depth.

Only base positions that fall under the user-specified coverage-region bed regions are present in the `_full_res.bed` output file.

The `_full_res.bed` file structure is similar to the output file of bedtools genomecov -bg. The contents are identical if the bedtools command line is executed after filtering out alignments with mapping quality value of 0, and possibly filtering by a target BED (if specified).

The following is an example of the contents of the `_full_res.bed` file:

```
chr1 121483984 121483985 10
chr1 121483985 121483986 9
chr1 121483986 121483989 8
chr1 121483989 121483991 7
chr1 121483991 121483992 6
chr1 121483992 121483993 4
chr1 121483993 121483994 2
chr1 121483994 121484039 1
chr1 121484039 121484043 2
chr1 121484043 121484048 3
chr1 121484048 121484050 7
chr1 121484050 121484051 11
chr1 121484051 121484052 17
chr1 121484052 121484053 149
chr1 121484053 121484054 323
chr1 121484054 121484055 2414
```

Coverage is reported for all positions specified by `qc-coverage-region-i`. Masked regions in the FASTA are not ignored.

Coverage Report

The DRAGEN `cov_report` report generates a `_cov_report.bed` file in a tab-delimited format full-coverage-report file in both BED and CSV formats. The first three columns are standard BED fields. The remaining column fields are statistics calculated over the interval region specified on the same record line.

The following table lists the appended columns.

Column	Description
<code>total_cvg</code>	The total coverage value.
<code>mean_cvg</code>	The mean coverage value.
<code>Q1_cv</code>	The lower quartile (25th percentile) coverage value.

Column	Description
median_cvg	The median coverage value.
Q3_cvg	The upper quartile (75th percentile) coverage value.
min_cvg	The minimum coverage value.
max_cvg	The maximum coverage value.
pct_above_X	Indicates the percentage of bases over the specified interval region that had a depth coverage greater than X.

By default, if an interval has a total coverage of 0, then the record is written to the output file. To filter out intervals with zero coverage, set `vc-emit-zero-coverage-intervals` to false in the configuration file.

By default, if `--qc-coverage-region-#-thresholds` are not set, the thresholds will default to 5, 15, 20, 30, 50, 100, 200, 300, 400, 500, 1000.

The following is an example of the contents of the `_cov_report.bed` file:

chrom	start	end	total_cvg	mean_cvg	Q1_cvg	median_cvg	Q3_cvg	min_cvg	max_cvg
pct_above_5 ...									
chr5 34190121	34191570	76636	52.89	44.00	54.00	60.00	32	76	
100.00 ...									
chr5 34191751	34192380	39994	63.58	57.00	61.00	69.00	50	85	
100.00 ...									
chr5 34192440	34192642	10074	49.87	47.00	49.00	51.00	44	62	
100.00 ...									
chr9 66456991	66457682	31926	46.20	39.00	45.00	52.00	27	65	
100.00 ...									
chr9 68426500	68426601	4870	48.22	42.00	48.00	54.00	39	58	
100.00 ...									
chr17 41465818	41466180	24470	67.60	4.00	66.00	124.00	2	153	
66.30 ...									
chr20 29652081	29652203	5738	47.03	40.00	49.00	52.00	34	58	
100.00 ...									
chr21 9826182	9826283	4160	41.19	23.00	52.00	58.00	5	60	
99.01 ...									

Coverage/Callability Reports Use Cases and Expected Output

The following table shows which outputs are generated when default options (`--vc-target-bed`) versus optional coverage region options (`--coverage-region`) are used.

--VC- target-bed specified? Y/N	--qc- coverage-region-i (i equal to 1, 2, or 3) specified? Y/N	Expected Output Files
N	N	wgs_coverage_metrics.csv wgs_fine_hist.csv wgs_hist.csv wgs_overall_mean_cov.csv wgs_contig_mean_cov.csv
N	Y	wgs_coverage_metrics.csv wgs_fine_hist.csv wgs_hist.csv wgs_overall_mean_cov.csv wgs_contig_mean_cov.csv
For each coverage region specified by the user: qc-coverage-region-i_coverage_metrics.csv qc-coverage-region-i_fine_hist.csv qc-coverage-region-i_hist.csv qc-coverage-region-i_overall_mean_cov.csv qc-coverage-region-i_contig_mean_cov.csv qc-coverage-region-i_full_res.bed if full_res report type is requested for qc-coverage-region-i qc-coverage-region-i_cov_report.bed if cov_report report type is requested for qc-coverage-region-i qc-coverage-region-i_callability.bed if GVCF mode is enabled and the callability or exome-callability report type is requested		

--vc- target-bed specified? Y/N	--qc- coverage-region-i (i equal to 1, 2, or 3) specified? Y/N	Expected Output Files
Y	N	wgs_coverage_metrics.csv wgs_fine_hist.csv wgs_hist.csv wgs_overall_mean_cov.csv wgs_contig_mean_cov.csv
		target_bed_coverage_metrics.csv target_bed_fine_hist.csv target_bed_hist.csv target_bed_overall_mean_cov.csv target_bed_contig_mean_cov.csv
		target_bed_callability.bed if GVCF mode is enabled

--VC- target-bed specified? Y/N	--qc- coverage-region-i (i equal to 1, 2, or 3) specified? Y/N	Expected Output Files
Y	Y	wgs_coverage_metrics.csv wgs_fine_hist.csv wgs_hist.csv wgs_overall_mean_cov.csv wgs_contig_mean_cov.csv
		target_bed_coverage_metrics.csv target_bed_fine_hist.csv target_bed_hist.csv target_bed_overall_mean_cov.csv target_bed_contig_mean_cov.csv target_bed_callability.bed if GVCF mode is enabled
		For each coverage region specified by the user: qc-coverage-region-i_coverage_metrics.csv qc-coverage-region-i_fine_hist.csv qc-coverage-region-i_hist.csv qc-coverage-region-i_overall_mean_cov.csv qc-coverage-region-i_contig_mean_cov.csv
		qc-coverage-region-i_full_res.bed if full_res report type is requested for qc-coverage-region-i qc-coverage-region-i_cov_report.bed if cov_report report type is requested for qc-coverage-region-i qc-coverage-region-i_callability.bed if GVCF mode is enabled and the callability or exome-callability report type is requested

BigWig Compression of Coverage Metrics

When `--enable-metrics-compression` is set to true, the 1 bp resolution coverage metrics output bed file (`_full_res.bed`) are compressed to bigwig format.

Read Coverage Report

The `read_cov_report` generates a `_read_cov_report.bed` file in a tab-delimited format. The first five columns are `chrom`, `start`, `end`, `name`, and `gene_id` BED fields. The following additional columns represent statistics that are calculated over the interval region specified on the same record line.

- `total_cvg`—The total coverage value.

- `read1_cvg`—The total Read 1 coverage value.
- `read2_cvg`—The total Read 2 coverage value.

If an alignment overlaps more than one region, the alignment is counted toward the region with the largest overlap. If the alignment overlaps equally with more than one region, the alignment is counted toward the first intersecting region.

The following example shows the contents of the `_read_cov_report.bed` file.

```
#chrom start end name gene_id total_cvg read1_cvg read2_cvg
chr21 10033000 10034919 48 24 24
chr21 10034919 10034920 0 0 0
chr21 10034920 10034921 0 0 0
```

GC Bias Report

The GC bias report provides information on GC content and the associated read coverage across a genome. DRAGEN GC bias metric is modeled after the Picard implementation and adapted to preexisting internal measures. The DRAGEN GC bias correction module attempts to correct these biases following the target count stage. For more information, see [GC Bias Correction on page 186](#)

The GC bias metric is computed as follows.

1. Calculates GC content using a 100 bp wide, per-base rolling window over all chromosomes in the reference genome, excluding any decoys and alternate contigs. Windows containing more than four masked (N) bases in the reference are discarded.
2. Calculates the average coverage for each window, excluding any non-PF, duplicate, secondary, and supplementary reads.
3. Calculates the average global coverage across the whole genome.
4. Groups valid windows based on the percentage of GC content, both at individual percentages and five 20% ranges as summary.
5. Calculates the normalized coverage for each group by dividing the average coverage for the bin by the global average coverage across the genome. Values below 1.0 indicate a lower than expected coverage at the given GC percent or range. Coverages significantly deviating from 1.0 at greater GC values are an expected result.
6. Calculates dropout metrics as the sum of all positive values of (percentage of windows at GC X- percentage aligned reads at GC X) for each GC $\leq 50\%$ and $> 50\%$ for AT and GC dropout.

By default, the GC bias metric report is not calculated. To enable GC Bias calculations, enter the `--gc-metrics-enable` command line option. The following is an example command:

```
$ dragen -b <BAM file> -r <reference genome> --gc-metrics-enable=true
```

The GC metrics report generates a `gc_metrics.csv` file. The file is structured as follows.

```
GC BIAS DETAILS,,Windows at GC [0-100],<number of windows>,<fraction of all windows>
```

```

GC BIAS DETAILS,,Normalized coverage at GC [0-100],<average coverage of all windows at given GC divided by average coverage of whole genome>
GC METRICS SUMMARY,,Window size,<window size in base, typically 100>
GC METRICS SUMMARY,,Number of valid windows,<total number of windows used in calculations>
GC METRICS SUMMARY,,Number of discarded windows,<total number windows discarded due to more than 4 masked bases>
GC METRICS SUMMARY,,Average reference GC,<average GC content over all valid windows>
GC METRICS SUMMARY,,Mean global coverage,<average genome coverage over all valid windows>
GC METRICS SUMMARY,,Normalized coverage at GCs <GC range>,<average coverage of all windows at given GC range divided by average coverage of whole genome>
GC METRICS SUMMARY,,AT Dropout,<Calculated AT dropout value>
GC METRICS SUMMARY,,GC Dropout,<Calculated GC dropout value>

```

The GC bias report also includes the following command line options, but they are not recommended.

- `--gc-metrics-window-size`—Overrides the default rolling window size of 100 bp.
- `--gc-metrics-num-bins`—Overrides the number of summary bins.

Somatic Metrics Report

In somatic tumor-normal mode, DRAGEN generates separate reports for the tumor and normal samples. Each report is labeled according to the sample type. Tumor sample reports include `tumor` at the end of the file name, and normal sample reports include `normal` at the end of the file name. To include both tumor and normal sample results in one file, set the `--vc-enable-separate-t-n-metrics` option to false. DRAGEN then reports on the aggregate of both samples instead.

DRAGEN generates the following reports:

- `coverage_metrics`
- `contig_mean_cov`
- `fine_hist`
- `hist`
- `overall_mean_cov`
- `ploidy`
- `cov_report` (if requested)
- `full_res` (if requested)
- `gc_metrics` (if requested)

If a target bed is included in the run or the option `--qc-coverage-region-i` (where *i* is 1, 2, or 3) is included, then the corresponding coverage reports are also split into tumor and normal files.

In somatic tumor-only mode, the reports are identical to those in germline mode and do not include tumor or normal in the file names.

Somatic Callable Regions Report

In somatic mode, DRAGEN automatically generates a somatic callable regions report as a bed file. The somatic callable regions report includes all regions with tumor coverage at least as high as the tumor threshold and (if applicable) normal coverage at least as high as the normal threshold. If only the tumor sample is provided, then the report includes all regions with tumor coverage at least as high as the tumor threshold. Each line in the bed output file is formatted as follows.

```
chromosome region_start region_end
```

You can specify the threshold values using the `--vc-callability-tumor-thresh` or `--vc-callability-normal-thresh` options. The default value for the tumor threshold is 15. The default value for the normal threshold is 5. For more information on each option, see [Somatic Mode Options on page 127](#).

If the target bed or the `--qc-coverage-region-i` (where i is 1, 2, or 3) option is included in the run, DRAGEN then generates corresponding somatic callable regions bed files in addition to the whole genome somatic callable region bed file.

DRAGEN HLA Caller

DRAGEN includes a dedicated human leukocyte antigen (HLA) genotyper for calling HLA class I and class II alleles with two-field resolution, also known as four-digit resolution. At this resolution, DRAGEN HLA genotyper is able to discern and report HLA alleles based on their protein sequences. For more information on HLA nomenclature, see *Nomenclature for factors of the HLA system*¹

Class I HLA typing is enabled by setting `--enable-hla` flag to `true`. Additionally, class II HLA typing is enabled by setting the `--hla-enable-class-2` flag to `true`. For TSO500-solid or TSO500-liquid runs, HLA typing should be enabled instead through the following batch options: `--tso500-solid-hla=true` and `--tso500-liquid-hla=true` respectively. Class II HLA typing is not supported for TSO500 runs.

¹ MarshSG, et al. Nomenclature for factors of the HLA system, 2010. *Tissue Antigens*. 2010 75:291-455.

Workflow

The HLA Caller primarily executes the following four steps::

1. Extract reads mapped to the HLA genes. These are HLA-A, -B and -C loci for class I, and HLA-DQA1, -DQB1, -DRB1 for class II loci. The human reference version is auto-detected during this step. The human reference builds hg19, hs37d5, and GRCh38 are fully supported, CHM13 build is enabled but not supported.

2. Align the extracted HLA reads to a reference set of 9,086 HLA alleles using the DRAGEN map-align processor. Only full-sequence alleles from the IMGT/HLA database (v3.45) that have also been reported on the Allele Frequency Net database were selected in building the default HLA reference resource.
3. Filter out HLA-specific alignments with sub-maximal alignment scores, and optimize the read distribution using Expectation-Maximization.
4. Select the most likely genotype for each HLA locus from a short list of candidate alleles using a homozygosity threshold set at 20%.

The following example command enables class I and II HLA typing for a run from input FASTQ.

```
dragen \
--enable-hla=true \
--hla-enable-class-2=true \
--output-directory={output_directory} \
--output-file-prefix={prefix} \
--enable-map-align=true \
--RGID=read_group_ID \
--RGSM=read_group_sample \
--ref-dir={reference_directory} \
-1 {fq1} \
-2 {fq2}
```

Reference Requirement for HLA

The reference directory that is supplied at command-line with `--ref-dir` must contain `anchored_hla`, a specific subdirectory with HLA-specific reference files. The DRAGEN Compute Server default reference directories have been updated to contain the `anchored_hla` subdirectory.

Building the HLA-Specific Reference Subdirectory

An HLA-specific reference subdirectory can be built by executing

```
dragen \
--build-hash-table true \
--ht-build-hla-hashtable=true \
--output-directory={REF-DIR}
```

This command will create `anchored_hla` as a subdirectory of the target `{REF-DIR}` supplied as an argument to `--output-directory` as above.

The HLA-specific reference subdirectory can be built at the same time as the primary reference construction. An example command-line for this mode is

```
dragen \
--build-hash-table true \
--ht-build-hla-hashtable=true \
--output-directory={REF-DIR} \
--ht-reference {PATH-TO}primary_reference.fasta
```

Resource FASTA

An HLA resource file, `HLA_resource.v2.fasta.gz`, is packaged with DRAGEN and is located at `/opt/edico/resources/hla/HLA_resource.v2.fasta.gz`.

The file is used by default when building the HLA-specific hash-table, see [Building the HLA-Specific Reference Subdirectory](#) for more information.

Using Custom HLA Reference Files

An HLA allele reference FASTA file can be used as input to the hash-table building option `--ht-hla-reference`.

 | Using custom HLA reference files to generate the HLA-specific reference subdirectory `anchored_hla` is not recommended, as accuracy cannot be guaranteed.

Custom input FASTA files (which can be zipped or unzipped) must contain only HLA allele sequences, and all allele names must adhere to the HLA star-allele nomenclature¹, where the first character of each allele name indicates the HLA locus, eg `A*02:01:01:01`. Allele names extracted from such a custom input file start at the first character of the allele name (to be preceded by character '`>`') and end at the last character of the name or until the first delimiter character '`-`' is reached.

The following is an illustration of a valid HLA reference input file to option `--ht-hla-reference`:

```
>A*01:01:01:30-full
TCCCCAGACGCCGAGGATGGCCGTATGGCGCC...
>A*01:01:01:47-full
TCCCCAGACGCCGAGGATGGCCGTATGGCGCC...
>A*01:01:01:76-full
TCCCATTGGGTGTCGGGTTCCAGAGAACCAA...
>A*01:01:01:91-full
TCCCCAGACGCCGAGGATGGCCGTATGGCGCC...
...
...
```

Custom HLA reference files might require customized memory allocation, which can be specified with an argument to the command line option `--ht-hla-ext-table-alloc`.

Pipeline Options

The HLA component has no additional user-settable command line options.

The HLA component replaces prior workflows. See the appropriate guide for the DRAGEN software version being used to determine valid parameters.

Map-Align DRAGEN Requirement for HLA

The HLA Caller requires the DRAGEN mapper-aligner to be enabled (enabled via option `--enable-map-align=true`, or through TSO500-batch options).

Output Files

The HLA Caller generates a tab-delimited output file for class I and, if enabled, class II alleles. Class I results contain six class I alleles, with two alleles per class I HLA gene (HLA-A, -B and -C), and class II results contain six class II alleles, with two alleles per class II HLA gene (HLA-DQA1, -DQB1, and -DRB1). Homozygous calls show identical alleles at the respective loci.

The genotype output file is `<prefix>.hla.tsv`, and it is located in the user-specified output directory. In tumor-only mode the output is stored to `<prefix>.hla.tumor.tsv` file. In tumor-normal mode, two output genotype files are generated from tumor and normal samples: `<prefix>.hla.tumor.tsv` and `<prefix>.hla.tsv`.

In all cases, the genotype file contains a header row with one column for each of the class I and/or class II alleles and a body row with the HLA type of each allele at two-field resolution.

The following is an example output file produced by DRAGEN class I and II HLA typing:

A1	A2	B1	B2	C1	C2	DQA11	DQA1 2	DQB11	DQB12	DRB11	DRB1 2
A*2	A*29	B*44	B*44	C*0	C*16	DQA1*0	DQA1*0	DQB1*0	DQB1*0	DRB1*1	DRB1*1
6:01	:02	:02	:03	5:01	:01	1:03	1:02	6:03	6:02	5:01	5:01

The HLA Caller generates two additional HLA files.

- `<prefix>.hla_metrics.csv` Contains the number of reads supporting each allele result (individual reads may support multiple alleles), and the total number of HLA reads analyzed.
- `<prefix>.hla_2field_EM.tsv` Contains the maximal likelihood output from the Expectation-Maximization step: a list of candidate alleles at two-field resolution and corresponding intermediate posterior probability.

Internal checks for sufficient coverage at each HLA locus will trigger a warning message when fewer than 50 reads support any given allele call, or when fewer than 300 HLA reads are detected overall. In both settings, an allele call will still be attempted, but the results may be unreliable.

An empty genotype call at a given HLA locus is returned when there are no reads supporting that locus. In this scenario, a warning message will indicate missing coverage.

Known Limitations

- Map-align must be enabled for HLA (see Map-Align DRAGEN Requirement for HLA). Tumor-normal paired file inputs from BAM are not currently supported for HLA calling.
- No HLA genotype will be returned with single-end DNA read inputs.
- By default, DRAGEN only genotypes HLA alleles that have full-nucleotide sequence data in IMGT/HLA v3.45 and that have also been reported on the Allele Frequency Net database. Partial alleles are not currently called using the supplied resource reference FASTA file `HLA_resource.v2.fasta`.

Examples

The HLA Caller accepts standard input files in FASTQ and BAM format.

The following example command line uses FASTQ file inputs.

```
dragen \
--enable-hla=true \
--enable-map-align=true \
--enable-sort=true \
--output-directory={output_directory} \
--output-file-prefix={prefix} \
--ref-dir={reference_directory} \
--RGID={read_group_ID} \
--RGSM={read_group_sample} \
-1 {fq1} \
-2 {fq2}
```

The following example command line uses BAM file inputs (with map-align enabled).

```
dragen \
--enable-hla=true \
--enable-map-align=true \
--enable-sort=true \
--output-directory={output_directory} \
--output-file-prefix={prefix} \
--bam-input={bam} \
--ref-dir={reference_directory} \
```

The following example command line uses tumor-normal paired file inputs from FASTQ.

The `--hla-enable-class-2` enables class II HLA typing.

```
dragen \
--enable-hla=true \
--hla-enable-class-2=true \
--enable-map-align=true \
--enable-sort=true \
--output-directory={output_directory} \
--output-file-prefix={prefix} \
--ref-dir={reference_directory} \
--tumor-fastq1={tumor_fq1} \
--tumor-fastq2={tumor_fq2} \
--RGID=tumor={tumor_group_ID} \
--RGSM=tumor={tumor_group_sample} \
-1 {normal_fq1} \
-2 {normal_fq2} \
--RGID={normal_group_ID} \
--RGSM={normal_group_sample} \
```

The following example command line activates HLA typing in a TSO500-solid run from FASTQ input.

```
dragen \
--tso500-solid-umi=true \
--tso500-solid-hla=true \
--fastq-file1={tumor_fq1} \
--fastq-file2={tumor_fq2} \
--RGID={read_group_ID} \
--RGSM={read_group_sample} \
--ref-dir={TSO500-compatible_reference_directory} \
--output-directory={output_directory} \
--output-file-prefix={prefix}
```

The following example command line activates HLA typing in a TSO500-liquid run from FASTQ input.

```
dragen \
--tso500-liquid=true \
--tso500-liquid-hla=true \
--fastq-file1={tumor_fq1} \
--fastq-file2={tumor_fq2} \
--RGID={read_group_ID} \
--RGSM={read_group_sample} \
```

```
--ref-dir={TSO500-compatible reference_directory} \
--output-directory={output_directory} \
--output-file-prefix={prefix}
```

Biomarkers

Tumor Mutational Burden

DRAGEN supports Tumor Mutational Burden (TMB) in Tumor-Only or Tumor-Normal Mode.

It is important to note that in T/O mode germline variants must be identified and filtered using database information and optionally also allele frequency information. These germline filtering techniques are generally not as accurate as tumor normal subtraction. When using databases only to subtract germline variants, the TMB may be slightly higher than the more accurate T/N estimate. When using database and allele frequency information to remove germline variants, the TMB may be slightly underestimated for high purity tumor samples.

DRAGEN TMB comprise the following steps:

1. Variant calling

Please refer to "Somatic mode" for detailed variant calling options.

2. Eligible region detection

The user specifies regions of interests (eg coding regions) with qc-coverage-region1. The subset of these regions that also exceed the specified minimum required coverage vc-callability-tumor-thresh are identified as eligible.

3. Variant filters

The following variants are excluded from the TMB calculation:

- Non-PASS variants
- Mitochondrial variants
- MNVs
- Variants that do not meet the minimum depth (DP) threshold. Use the --vc-callability-tumor-thresh command line option to specify the threshold value.
- Variants that do not meet the minimum variant allele threshold. Use the --tmb-vaf-threshold command line option to specify the threshold value.
- Variants that fall outside the eligible regions.
- Tumor driver mutations. Variants with a population allele count ≥ 50 are treated as tumor driver mutations. You can specify the cosmic driver threshold using the tmb-cosmic-count-threshold command line option. The tumor driver mutations filter relies on Nirvana annotations and will additionally require settings for --enable-variant-annotation=true, --variant-annotation-assembly, and --variant-annotation-data.

4. Support for germline variants

By default, germline variants are not counted towards TMB. Variants are determined as germline based on a database or a proxy filter. The database germline filter can be disabled with `tmb-skip-db-filter`. Disabling the database germline filter will effectively also disable the germline proxy filter.

Database filter

Variants with a population allele count ≥ 50 that are observed in either the 1000 Genome or gnomAD database will be marked as germline. Use `germline-tagging-db-threshold` to change the population allele counts. The database germline filter relies on Nirvana annotations and requires settings for `--enable-variant-annotation=true`, `--variant-annotation-assembly`, and `--variant-annotation-data`.

Proxy filter

- Proxy filter can be enabled with `tmb-enable-proxy-filter`. The proxy filter will flag any variants with VAF > 0.9 as germline. The proxy filter scans the variants around a specific variant and identifies those variants with similar VAFs. The proxy window size that determines the number of surrounding variants can be specified with `tmb-proxy-window-size`. If 95% (default value for `tmb-proxy-fraction-threshold`) and no less than 5 (`tmb-proxy-count-threshold`) of the surrounding variants of similar VAF are germline, then mark the current variant also as germline.
- Proxy filter can also be done via a probabilistic approach, which can be enabled with `tmb-enable-prob-proxy`. It estimates the expected germline allele frequency using the surrounding germline variants and then tests whether the allele frequency of the target variant is similar to the expected germline allele frequency or not. P value threshold can be set by `tmb-prob-proxy-p-value` (the default value $1e-15$ is set for ultra-deep sequenced samples, e.g cfDNA)
- Note that proxy filters can be too aggressive for 100% pure cell lines. Probabilistic proxy filter can be problematic for mixing or contaminated samples, as these samples do not have clear germline variant allele frequency distributions.

CH filter

When processing ctDNA samples it may be beneficial to also remove CH (clonal hematopoiesis) variants. Circulating tumor DNA generally has shorter fragment size. CH variants can be identified based on the insert size of the reads supporting the call. To capture the insert-size distribution for each variant call, it is required to specify `vc-log-insert-size` during variant calling (step1). Once specified, potential CH variants based on insert size distribution will be labeled in the output. Additional, CH variants can be also labeled via a bed file supplied to `tmb-ch-bed`. Variants other than germlines overlapping the region will be labeled as CH.

5. Support Nonsynonymous variants.

Nonsynonymous consequences are detected based on the Nirvana annotations. Nirvana variants that are annotated with the following consequences are labeled as nonsynonymous:

- feature_elongation, feature_truncation, frameshift_variant, incomplete_terminal_codon_variant, inframe_deletion, inframe_insertion, missense_variant, protein_altering_variant, splice_acceptor_variant, splice_donor_variant, start_lost, stop_gained, stop_lost, transcript_truncation

TMB emits a `tmb.trace.csv` file with detailed information on each variant used the TMB score. The trace file contains a column "Nonsynonymous" that indicates the appropriate status for each variant.

The subset of filtered variants that are nonsynonymous are used as numerator in the "Filtered Nonsyn Variant Count" metric.

6. TMB calculation

$TMB = \text{Filtered Variants} / \text{Eligible Region (Mbp)}$

$\text{Nonsynonymous TMB} = \text{Filtered Nonsynonymous Variants} / \text{Eligible Region (Mbp)}$

7. Maximum somatic allele frequency (MSAF)

The maximum somatic allele frequency (MSAF) outputs the estimated maximum somatic allele frequency of the sample. This is done via finding the confident somatic variants with highest allele frequency. MSAF is a rough approximate to the tumor fraction of cfDNA in peripheral blood samples.

The MSAF mode can be enabled with `tmb-enable-msaf`.

Command-Line Options

Required

Command-line Option	Description
<code>--enable-tmb true</code>	Enables TMB. If set, the small variant caller, DRAGEN Annotation Engine, and the related callability report are enabled.
<code>--qc-coverage-tag-1</code>	Set <code>--qc-coverage-tag-1=tmb</code> to execute DRAGEN TMB.
<code>--qc-coverage-region-1</code>	Specify the coding regions to use.
<code>--vc-callability-tumor-thresh</code>	Set <code>--qc-coverage-tag-1=tmb</code> to execute DRAGEN TMB.
<code>--qc-coverage-reports-1=callability</code>	The callability report is required for DRAGEN TMB.
<code>--enable-variant-annotation=true</code>	Enables Nirvana, the Illumina Annotation Engine. For more information on selecting the correct assembly and downloading reference files, see Illumina Annotation Engine on page 653 .
<code>--variant-annotation-assembly</code>	
<code>--variant-annotation-data</code>	

Recommended

Setting	Description	Tumor-Normal Panel/WES/WGS	Tumor-Only Panel/WES/WGS	High coverage bTMB
--vc-callability-tumor-thresh	The minimum coverage for usable coding regions	50	50	1000
--tmb-vaf-threshold	Variant minimum allele frequency for usable variants	0.05	0.05	0.002
--tmb-cosmic-count-threshold	Number of observations in cosmic for variant to be considered a driver mutation.	50	50	50
--tmb-skip-db-filter	Do not use Nirvana database to filter germline variants	TRUE	FALSE	FALSE
--tmb-enable-proxi-filter	Use allele frequency information to filter germline variants	OPTIONAL (Default is FALSE)	FALSE	TRUE

Optional

Command-line Option	Description
--tmb-vaf-threshold	Specify the minimum VAF threshold for a variant. Variants that do not meet the threshold are filtered out (default=0.05)
--tmb-cosmic-count-threshold	The minimum number of observations in cosmic for variant to be considered a driver mutation. Driver mutations are not counted in TMB. This setting has very little impact on WES/WGS, but can help avoid bias in small panels (default=50)

Command-line Option	Description
--tmb-skip-db-filter	Skip database germline filtering. The database germline filter is required for tumor-only samples, but can be skipped for tumor-normal (default=false)
--germline-tagging-db-threshold	Specify the minimum allele count (total number of observations) for an allele in gnomAD or 1000 Genome to be considered a germline variant. Variant calls that have the same positions and allele are ignored from the TMB calculation (default=50)
--tmb-enable-proxi-filter	Enable proxி filter functionality in germline filtering. This is an optional feature that may be appropriate for T/O runs. In T/O mode the DB germline filter may not able to detect all germline variants, especially for ethnicity groups that are not well represented in germline databases. The proxி filter uses allele frequency information to help remove germline variants missed by the DB, and can help to obtain more accurate (lower) TMB values on samples with low tumor purity. In samples with high tumor purity this filter may be too aggressive and mark some somatic variants as germline resulting in too low TMB scores. (default=false)
--tmb-proxi-count-threshold	Proxi filter surrounding variant count threshold in germline filtering (default=5)
--tmb-proxi-fraction-threshold	Proxi filter surrounding variant db filter fraction threshold in germline filtering (default=0.95)
--tmb-proxi-window-size	Number of surrounding variants before and after the target variant for proxி filter (default=500)
--tmb-ch-bed	Variants in the region will be labeled as clonal hematopoiesis (CH) variants
--tmb-ch-insert-p-value	Minimum P value to classify a variant as CH using insert size (default=0.1)
--tmb-ch-insert-min-len	Minimum fragment size to test for CH using insert size (default=100)
--tmb-ch-insert-max-len	Maximum fragment size to test for CH using insert size (default=200)
--tmb-ch-insert-min-num	Minimum number of fragment size record to test for CH using insert size (default=50)
--tmb-enable-msaf	Enable MSAF output (default=false)

Command-line Option	Description
--tmb-msaf-p-value	Maximum P value (from insert size) to call confident somatic variant (default=1e-5)
--tmb-msaf-rank-num	If no confident somatic variant found, it will use the specified ranked variant (default=4)

Output

The TMB values are output to <output prefix>.tmb.metrics.csv. The file format uses the following CSV column convention, similar to other metric CSV files.

Metric	Definition
Eligible Region (Mbp)	The specified custom regions that meet the minimum coverage threshold.
Filtered Variant Count	Remaining variants after VAF, variant type, and germline filters.
Filtered Nonsyn Variant Count	Filtered variants that are filtered further to include only nonsynonymous events.
TMB	Filtered variants by the eligible regions. ?
Nonsyn TMB	Filtered nonsynonymous variants by the eligible regions.

The TMB module also emits a `tmb.trace.csv` file that provides detailed information on each variant that was included in the TMB calculation.

When enabling MSAF, the information is output to <output prefix>.tmb.msaf.csv.

Microsatellite Instability

Microsatellites are genomic regions of short DNA motifs that are repeated 5–50 times and are associated with high mutation rates. Microsatellite Instability (MSI) results from deficiencies in the DNA mismatch repair pathway and can be used as a critical biomarker to predict immunotherapy responses in multiple tumor types.

DRAGEN MSI can work in 3 different modes determined by enabling the option `--msi-command`:

- Collect Evidence mode (`collect-evidence`)
- Tumor Normal mode (`tumor-normal`)
- Tumor Only mode (`tumor-only`)

Three modes share some of the following steps:

1. Tabulates tumor and normal counts from the read alignments for each microsatellite site.

2. Calculates Jensen-Shannon distance of tumor and normal distribution for each microsatellite site (tumor-normal mode), or Jensen-Shannon distance of two normal baseline samples (tumor-only mode).
3. Determines unstable sites by performing chi-square testing of tumor and normal distribution. Unstable sites have repeat length distributions that are significantly shifted between tumor and normal measured by Jensen-Shannon distance (tumor-normal mode). In tumor-only mode, JSD is calculated for each pair of tumor and normal reference samples, as well as each pair of normal-normal samples. Then the two sets of JSD is compared to derive a mean distance difference and p-value calculated from student t-test. Microsatellite instability is called if the mean distance difference is greater than or equal to the distance threshold (default 0.1) and p-value less than or equal to the p-value threshold (default 0.01).
4. Produces a report given aassessed site count, unstable site count, the percentage of unstable sites in all assessed sites and the sum of the Jensen-Shannon distance of all the unstable sites.

`collect-evidence` mode only performs step 1 and reports the microsatellite distribution for each site in a given sample.

Command-Line Options

Use the following commands to compute MSI:

Command-Line Option	Description
<code>-msi-command tumor-only/tumor-normal/collect-evidence</code>	Mode of execution: tumor-only, tumor-normal, or collect-evidence.
<code>-msi-microsatellites-file</code>	Specify the file containing the microsatellites. You can generate this file by scanning the genome for microsatellites using an MSI-sensor. DRAGEN has tested with ≥ 10 bp homopolymers for solid samples, and 6-7 bp homopolymers for liquid samples..
<code>-msi-ref-normal-dir</code>	Full name of directory containing files with normal reference repeat length distribution. Used only in <code>tumor-only</code> mode. These files can be generated by running <code>collect-evidence</code> on each normal sample. A minimum of 20 normal samples is required for tumor-only mode.

Command-Line Option	Description
-msi-coverage-threshold	Specify the minimum spanning read coverage for a microsatellite. Microsatellites that do not meet the specified threshold are not included in analysis. DRAGEN recommends using 60 as the value for solid samples. For liquid samples, a value of 500 is recommended.
-msi-distance-threshold	Threshold for distance distributions to be considered different. Default is 0.1. For liquid samples, a value of 0.02 is recommended.

The following is an example of a microsatellite file:

```
#chromosome location repeat_unit_length repeat_unit_binary repeat_times left_
flank_binary right_flank_binary repeat_unit_bases left_flank_bases right_flank_
bases
chr1 985443 1 2 15 676 992 G GGGCA TTGAA
chr1 7980985 1 0 10 231 1020 A ATGCT TTTTA
chr1 8022800 1 3 19 13 41 T AAATC AAGGC
chr1 8029500 1 2 10 39 0 G AAGCT AAAAAA
chr1 9146447 1 3 15 887 248 T TCTCT ATTGA
chr1 9767837 1 3 12 704 195 T GTAAA ATAAT
```

For WGS or WES data, we recommend to run only `tumor-normal` mode as it is fully supported and tested. The following is an example command for `tumor-normal` mode, including default input files.

```
dragen \
--msi-command tumor-normal \
--msi-coverage-threshold 60 \
--msi-microsatellites-file msi_file \
--output-directory={output_directory} \
--output-file-prefix={prefix} \
--enable-map-align=true \
--RGID=read_group_ID \
--RGSM=read_group_sample \
--ref-dir={reference_directory} \
--enable-map-align-output=true \
--enable-sort=true \
--enable-duplicate-marking=true \
--tumor-fastq1 {tumor_fq1} \
--tumor-fastq2 {tumor_fq2} \
```

```
--fastq-file1 {fq1} \
--fastq-file2 {fq2}
```

TSO500 panels do not have normal controls, therefore, only `tumor-only` mode is supported. The following is an example command for `tumor-only` mode, including default input files.

```
dragen \
--msi-command tumor-only \
--msi-coverage-threshold 60 \
--msi-microsatellites-file msi_file \
--msi-ref-normal-dir normal_reference_directory \
--output-directory={output_directory} \
--output-file-prefix={prefix} \
--enable-map-align=true \
--RGID=read_group_ID \
--RGSM=read_group_sample \
--ref-dir={reference_directory} \
--enable-map-align-output=true \
--enable-sort=true \
--enable-duplicate-marking=true \
--tumor-fastq1 {tumor_fq1} \
--tumor-fastq2 {tumor_fq2}
```

Microsatellite sites file

Microsatellite sites file can be obtained by scanning the reference genome to look for microsatellite sites of interests. We recommend to use external tools such as `msi-sensor` [<https://github.com/xjtu-omics/msisensor-pro/wiki/Best-Practices>] to generate this file.

Normal references of miscrosatellite repeat distribution

Normal reference files can be generated by running `collect-evidence` mode on a panel of normal samples.

 This only works with DRAGEN germline mode.

MSI Output

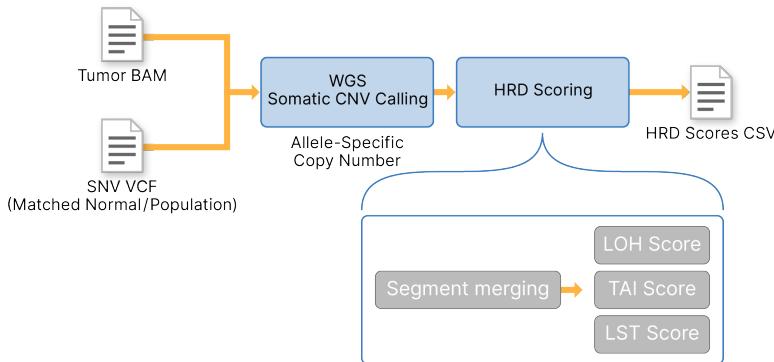
The output containing MSI score PercentageUnstableSites are output to <output prefix>.microsat_output.json.

Metric	Count
TotalMicrosatelliteSitesAssessed	20020
TotalMicrosatelliteSitesUnstable	4374
PercentageUnstableSites	21.850000000000001
ResultIsValid	true
ResultMessage	
SumDistance	1214.174

For solid samples, PercentageUnstableSites ≥ 20 indicates microsatellite instability. For liquid samples, sumJSD SumDistance ≥ 0.08 indicates microsatellite instability.

Homologous Recombination Deficiency

DRAGEN Homologous Recombination Deficiency (HRD) Scoring takes in allele-specific copy number calls in either VCF format or directly streamed from somatic copy number callers. DRAGEN HRD then calculates scores for Loss of Heterozygosity (LOH), Telomeric Allelic Imbalance (TAI), and Large-Scale State Transition (LST). The three scores are output to the `.hrdscore.csv` file. You can only use DRAGEN HRD when inputting results from WGS somatic CNV calling or WES T/N somatic CNV calling.



Command-Line Options

Use the following command-line options to run HRD scoring. You can run HRD scoring with somatic CNV calling or after using somatic CNV calling results.

To run HRD scoring together with somatic CNV calling, use the following options. See [Somatic CNV Calling on page 215](#) for more parameters.

- `--enable-hrd`—Set to true to enable HRD scoring to quantify genomic instability.
- `--enable-cnv`—Set to true to enable CNV calling to run together with HRD scoring.

To run HRD scoring after somatic CNV calling, use the following options:

- `--enable-hrd`—Set to true to enable HRD scoring to quantify genomic instability.

- **--hrd-input-ascn**—Specify the allele-specific copy number file (**cnv.vcf.gz*). The CNV VCF file should include REF calls for proper HRD segmentation. See the option **--cnv-enable-ref-calls** in the CNV section.
- **--hrd-input-tn**—Specify the tumor normalized bin count file (*.tn.tsv.gz).

HRD Output

The following metrics are included in the `.hrdscore.csv` output file. The following is an example output file.

Sample	LOH_Score	TAI_Score	LST_Score	HRD_Score
Sample	16	17	28	61

BRCA Within Gene Large Rearrangement (LR)

Large genomic rearrangements affecting one or more exons account for approximately 5~10% of all disease-causing mutations in BRCA1 and BRCA2 genes in patients with hereditary breast and ovarian cancer syndrome. DRAGEN LR can detect within gene large genomic rearrangements in tumor-only mode for targeted panels, for example TruSight Oncology 500. The performance has been verified for BRCA1/2 with TruSight Oncology 500 Assay.

Command-Line Options

Use the following command line options to run large rearrangement detection. The same cmd line options can be tested on other tumor-only pipelines.

--tso500-solid-brca-lr=true Set to true enable large rearrangement parameters. This is not limited to TruSight Oncology 500 Assay.

--cnv-normals-list Specify the panel of normal samples to measure intrinsic biases of the upstream processes to allow for proper normalization. To generate a panel of normals, see the example command line section below. The panel of normal samples should be well matched to the case sample under analysis.

--cnv-target-bed Specify the targeted regions of the panel.

--cnv-within-gene-lr-bed Specify the gene regions in BED format to do large rearrangement calling. Example file:

```
chr13 32889617 32973809 BRCA2
```

Example to generate panel of normal

Run the following command on each normal sample to generate `.target.counts.gc-corrected.gz` file.

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--enable-cnv true \
--cnv-enable-gcbias-correction true \
--cnv-enable-split-intervals true \
--cnv-target-bed <BED> \
--tumor-bam-input <BAM>
```

Put the path to the generated `.target.counts.gc-corrected.gz` files into a txt file. One file per line. This will be the file given to `--cnv-normals-list`.

Example command lines

```
dragen \
-r <HASHTABLE> \
--output-directory <OUTPUT> \
--output-file-prefix <SAMPLE> \
--tso500-solid-brca-lr true \
--cnv-normals-list <PON> \
--cnv-target-bed <BED> \
--cnv-within-gene-lr-bed <GENE.bed> \
--tumor-bam-input <BAM>
```

LR Output

The output file `.cnv.LR.json` contains the breakpoints detected for each specified gene region. The following is an example output file.

```
"Breakpoints": {
  "BRCA1": {
    "nSegs": "1",
    "segments": [
      {
        "id": "BRCA1.1",
        "chromosome": "chr17",
        "start": "41197309",
        "stop": "41276383",
        "nBin": "95",
        "segmentMean": "0.77611423479585684",
        "segmentMeanLog2": "-0.36565907927374325"
      }
    ]
  }
}
```

```
}

]

} ,

"BRCA2": {

  "nSegs": "3",

  "segments": [

    {

      "id": "BRCA2.1",

      "chromosome": "chr13",

      "start": "32890596",

      "stop": "32945239",

      "nBin": "61",

      "segmentMean": "0.80852624347115876",

      "segmentMeanLog2": "-0.3066334928504777"

    },

    {

      "id": "BRCA2.2",

      "chromosome": "chr13",

      "start": "32950805",

      "stop": "32954284",

      "nBin": "8",

      "segmentMean": "0.45378940514841132",

      "segmentMeanLog2": "-1.1399051688173489"

    },

    {

      "id": "BRCA2.3",

      "chromosome": "chr13",

      "start": "32956413",

      "stop": "32972909",

      "nBin": "10",

      "segmentMean": "0.890907164346186",

      "segmentMeanLog2": "-0.16665298919550192"

    }

  ]

}

}
```

LR Output

The output file `.cnv.LR.json` contains the breakpoints detected for each specified gene region. The following is an example output file.

```
"Breakpoints": {
  "BRCA1": {
    "nSegs": "1",
    "segments": [
      {
        "id": "BRCA1.1",
        "chromosome": "chr17",
        "start": "41197309",
        "stop": "41276383",
        "nBin": "95",
        "segmentMean": "0.77611423479585684",
        "segmentMeanLog2": "-0.36565907927374325"
      }
    ]
  },
  "BRCA2": {
    "nSegs": "3",
    "segments": [
      {
        "id": "BRCA2.1",
        "chromosome": "chr13",
        "start": "32890596",
        "stop": "32945239",
        "nBin": "61",
        "segmentMean": "0.80852624347115876",
        "segmentMeanLog2": "-0.3066334928504777"
      },
      {
        "id": "BRCA2.2",
        "chromosome": "chr13",
        "start": "32950805",
        "stop": "32954284",
        "nBin": "8",
        "segmentMean": "0.45378940514841132",
      }
    ]
  }
}
```

```

"segmentMeanLog2": "-1.1399051688173489"
},
{
"id": "BRCA2.3",
"chromosome": "chr13",
"start": "32956413",
"stop": "32972909",
"nBin": "10",
"segmentMean": "0.890907164346186",
"segmentMeanLog2": "-0.16665298919550192"
}
]
}
}

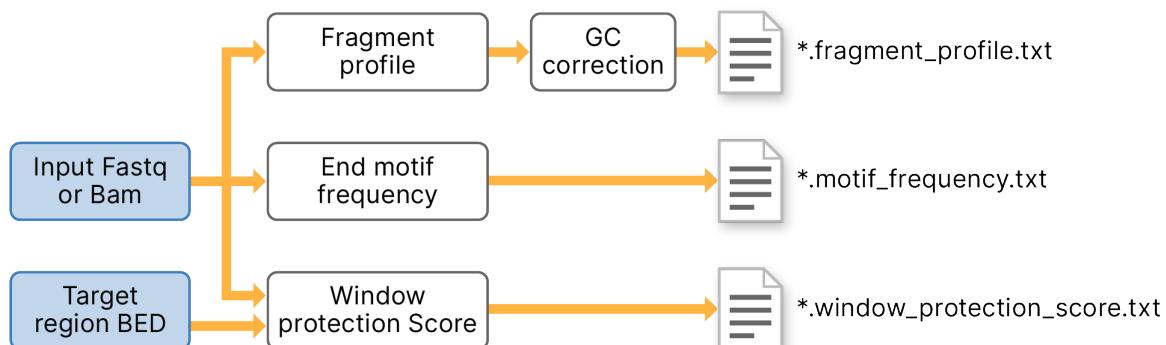
```

DRAGEN Fragmentomics

Fragmentomics is the study of fragmentation patterns of cell-free DNA or circulating tumor DNA (ctDNA). DNA molecules are released into the plasma from various tissues and cell types.

Fragmentation features, such as fragment sizes and end motifs, of the cell-free DNA contains the characteristics of their tissue of origin. Studies have shown that fragmentation features are distinct between cancer and noncancer cells derived from ctDNA. The use of genome-wide fragment profile of cell-free DNA has proven to be a powerful tool to infer cancer status and their tissue of origin. The DRAGEN fragmentomics component computes three fragmentomics metrics as following.¹

1. Fragment profile
2. End motif frequency
3. Window protection score (WPS)



The fragmentomics component works by taking aligned reads from the mapper, calculating per read metrics, and finally tabulating into per-bin or target region metrics. DRAGEN first gets the chromosome sizes from the reference genome. Only autosomes and X, Y chromosomes are considered for fragment profile calculation. The genome is binned with the bin size specified by the user. Each aligned read is processed sequentially. Only reads satisfied with the following criteria are considered:

- Mapped
- Mate-mapped
- Not a PCR duplicate
- Primary alignment
- Mapping quality is not less than minimum mapq specified by the user

Reads that have template length within the short fragment size ranges are counted as short fragment.

Reads that have template length within the long fragment size range are counted as long fragment. The fragment profile is calculated as the ratio of short-to-long fragment counts for each genomic bin.

Genome-wide short fragment counts, long fragment counts, and their ratio are normalized against the GC bias of each genomic bin using the GC correction module from DRAGEN CNV component.

End motif frequency calculation is enabled when `--fragmentomics-end-motif-len` is set to positive integers. Unmapped, duplicated, or secondary alignments are excluded for end motif frequency calculation. The first x base pair sequences (x is specified by `--fragmentomics-end-motif-len`) at the 5' end of the reads is tabulated into a frequency dictionary with keys being the sequences and values being the frequencies. If the first x basepair contains any 'N's, this read will be ignored. After all reads are processed, the frequency table is sorted by the sequences in alphabetic order.

Window protection score (WPS) calculation is enabled when a target region is provided with `--fragmentomics-wps-target-file`. This file must be a BED format text file with three columns. Each row in the file represents a 120-bp region for which WPS will be calculated. An interval tree will be constructed for the target regions. Then each aligned read is processed sequentially, and unmapped, duplicated, or secondary alignments are excluded. Any read with 5' end falling in a target region increments the read count for the region by one. Forward and reverse reads are counted separately. If a read fully spans the region, the fully-span read count for the region increments by one. After all reads are processed, WPS is calculated for each target region. Two ways of WPS calculation are supported, 1) number of fully spanning rads subtracted by the number of reads with 5' ending in the region. 2) percentage of reads ending in the region of all reads mapped to this region.

Supported assays and DRAGEN modes

DRAGEN Fragmentomics currently supports Tumor-only and Normal-only sequencing data from TSO500/WES/WGS ctDNA assays. The results for Tumor-Normal pair data are undefined because ctDNA data are derived from mixture of tumor and normal DNA. Therefore, users should avoid running Fragmentomics in Tumor-Normal mode.

Command Line Options

Required:

Enable the Fragmentomics component:

```
--enable-fragmentomics
```

Optional:

```
--fragmentomics-bin-size UInt. Default 100000
--fragmentomics-num-threads UInt. Default 12
--fragmentomics-min-mapq UInt. Default 30
--fragmentomics-short-fragment-min-size UInt. Default 100
--fragmentomics-short-fragment-max-size UInt. Default 150
--fragmentomics-long-fragment-min-size UInt. Default 151
--fragmentomics-long-fragment-max-size UInt. Default 220
--fragmentomics-num-gc-bins UInt. Default 25
--fragmentomics-gc-enable-smoothing Bool. Default true
--fragmentomics-end-motif-len UInt. Default 4
--fragmentomics-wps-target-file String
--fragmentomics-exclude-bed String
```

Target regions for window protection score

The target regions file is used only in window protection score calculation. The target regions file is in BED format with three columns.

```
chr1 2488101 2488120
chr1 2488120 2488174
```

Exclude regions for fragment profile

Users can provide a blocklist of regions to remove reads from fragment profile calculation. For example, low mappability regions. This file is in BED format with three columns.

```
chr1 1 1000
chr2 1000 2000
```

Example command line options for FASTQ input of WGS ctDNA

```
dragen \
--ref-dir=$REF \
```

```
--fastq-file1 $fastq1 \
--fastq-file2 $fastq2 \
--RGID "test" \
--RGSM "test" \
--enable-map-align true \
--enable-sort false \
--generate-ploidy-vcf false \
--enable-cnv false \
--enable-fragmentomics true \
--fragmentomics-exclude-bed hg19_exclude.bed \
--fragmentomics-bin-size 100000 \
--fragmentomics-num-threads 12 \
--fragmentomics-min-mapq 30 \
--fragmentomics-short-fragment-min-size 100 \
--fragmentomics-short-fragment-max-size 150 \
--fragmentomics-long-fragment-min-size 151 \
--fragmentomics-long-fragment-max-size 220 \
--fragmentomics-num-gc-bins 25 \
--fragmentomics-gc-enable-smoothing true \
--fragmentomics-end-motif-len 4 \
--fragmentomics-wps-target-file hg19.window.bed \
--output-directory $outdir \
--output-file-prefix $outprefix
```

Fragmentomics Output

The system should output the fragment profile file, and optionally the end motif frequency file or WPS file if either or both are enabled.

The fragment profile file is in the following format:

Chr	Start	End	ShortFragCounts ShortFragCountsCorrected	LongFragCounts LongFragCountsCorrected	ShortToLongRatio ShortToLongRatioCorrected	GCBias
chr1	0	100000	4 042	7 042	0.571429 0.571429	0.424522 0.424522
chr1	100000	200000	1 537	2 537	0.500000 0.500000	0.436106 0.436106

chr1 000	200000	300000	0	2	0.000000	0.391445	0.000000	2.008326	0.000
-------------	--------	--------	---	---	----------	----------	----------	----------	-------

The end motif frequency file is in the following format:

Motif	Frequency
AAAA	111559
AAAC	39204
AAAG	56773
AAAT	68437

The WPS file is in the following format:

Chr nt	Start RatioForward	End RatioReverse	ForwardCount	ReverseCount	FullySpanCount	WPS	TotalCou		
chr1	2488101	2488120	3	21	314	290	355	0.0084507	0.0591549
chr1	2488101	2488140	3	32	285	250	366	0.00819672	0.0874317
chr1	2488101	2488160	4	41	255	210	376	0.0106383	0.109043

1. Y. M. DENNIS LO, DIANA S. C. HAN, PEIYONG JIANG, ROSSA W. K. CHIU. Epigenetics, fragmentomics, and topology of cell-free DNA in liquid biopsies. *Science*. 2021. DOI: 10.1126/science.aaw3616

Downsampling

DRAGEN can reserve a random subset of reads that are separate from the normal alignment outputs using downsampling. You can use downsampling to generate data sets for performing comparisons between samples or between replicates. DRAGEN samples reads after performing any hardware accelerated trimming or filtering functions, which enables DRAGEN to rapidly create analysis-read test data sets.

To enable downsampling, set the `--enable-down-sampler` command line option to `true`.

You can use any valid sequencing data format that is compatible with the DRAGEN Host Software. For more information on compatible input options, see [Input Options on page 61](#).

DRAGEN downsampling outputs the reserved subset of data in FASTQ format. If the input is paired-ended, DRAGEN outputs two FASTQ files that contain subsampled data. If the input is unpaired, DRAGEN outputs two FASTQ files.

Command-line Settings

In addition to enabling the downsampling command line option, you must set the quantity of reads to downsample. To set the quantity of reads, use either `--down-sampler-reads` or `--down-sampler-coverage`.

If you specified a coverage level, you must also specify a genome using the `--ref-dir` or manually specify the genome size using `--down-sampler-genome-size`. If you specify both a read and coverage limit, DRAGEN applies both quantity limits and keeps whichever result is smaller.

Option	Description
<code>--enable-down-sampler</code>	Set to <code>true</code> to enable downsampling. The default value is <code>false</code> . If enabled, you must set either <code>down-sampler-reads</code> or <code>--down-sampler-coverage</code> .
<code>--down-sampler-num-threads</code>	Specify the number of threads to use for down-sampled reads. The default value is 8.
<code>--down-sampler-random-seed</code>	Set random seed for down-sampled reads. The default value is 42.
<code>--down-sampler-genome-size</code>	Set target genome size for downsampling coverage. The default value is 0. The <code>--down-sampler-genome-size</code> option is not compatible with the <code>--ref-dir</code> option.
<code>--down-sampler-reads</code>	Specify the target number of reads for downsampling. The default value is 0.
<code>--down-sampler-coverage</code>	Set target genomic coverage for downsampling. The default value is 0. If enabled, you must set either <code>-ref-dir</code> or <code>--down-sampler-genome-size</code> .

Virtual Long Read Detection

DRAGEN Virtual Long Read Detection (VLRD) is an alternate and more accurate variant caller focused on processing homologous/similar regions of the genome. A conventional variant caller relies on the mapper/aligner to determine which reads likely originated from a given location. It also detects the underlying sequence at that location independently of other regions not immediately adjacent to it. Conventional variant calling works well when the region of interest does not resemble any other region of the genome over the span of a single read (or a pair of reads for paired-end sequencing).

However, a significant fraction of the human genome does not meet this criterion. Many regions of the genome have near-identical copies elsewhere, and as a result, the true source location of a read might be subject to considerable uncertainty. If a group of reads is mapped with low confidence, a typical variant caller might ignore the reads, even though they contain useful information. If a read is mismapped (ie, the primary alignment is not the true source of the read), it can result in detection errors. Short-read sequencing technologies are especially susceptible to these problems. Long-read sequencing can mitigate these problems, but it typically has much higher cost and/or higher error rates, or other shortcomings.

DRAGEN VLRD attempts to tackle the complexities presented by the genome's redundancy from a perspective driven by the short-read data. Instead of considering each region in isolation, VLRD considers all locations from which a group of reads may have originated and attempts to detect the underlying sequences jointly using all available information.

Running DRAGEN VLRD

Like the DRAGEN variant caller, VLRD takes either FASTQ or sorted BAM files as input and produces an output VCF file. VLRD supports processing a set of only two homologous regions.

VLRD is not enabled by default. To run VLRD, set the `--enable-vlrd` option to true. The following is an example DRAGEN command to run VLRD.

```
dragen \
-r <REF> \
-1 <FQ1> \
-2 <FQ2> \
--RGID <RG> --RGSM <SM> \
--output-dir <OUTPUT> \
--output-file-prefix <PREFIX> \
--enable-map-align true \
--enable-sort=true \
--enable-duplicate-marking true \
--enable-vlrd true \
--vc-target-bed similar_regions.bed
```

VLRD Updated Map/Align Output

DRAGEN VLRD can output a remapped BAM/SAM file in addition to the regular DRAGEN Map/Align output. To enable output of a remapped BAM/SAM file, set the `--enable-vlrd-map-align-output` option to true. The default for this option is false.

The additional map/align output by VLRD only contains reads mapped to the regions that were processed by VLRD.

VLRD considers the information available from all the homologous regions jointly to update the read alignments. The updated VLRD map/align output is primarily useful for pile-up analysis involving homologous regions.

VLRD Settings

The following options are specific to VLRD in the DRAGEN host software.

Option	Description
<code>--enable-vlrd</code>	If set to true, VLRD is enabled for the DRAGEN pipeline.

Option	Description
--vc-target-bed	<p>Specifies the input bed file. DRAGEN requires an input target bed file specifying the homologous regions to be processed by VLRD. The input bed file id formatted to correctly process the homologous regions. The maximum region span processed by VLRD is 900 bp.</p> <p>For example:</p> <pre>chr1 161497562 161498362 0 0 chr1 161579204 161580004 0 0 chr1 21750837 21751637 1 0 chr1 21809355 21810155 1 1</pre> <ul style="list-style-type: none"> The first three columns are like traditional bed files: column 1 is chromosome description, column 2 is region start, and column 3 is region end. Column 4 is homologous region Group ID. This groups regions that are homologous to each other. Rows 1 and 2 have the same value in column 4 indicating that these should be processed as a set of homologous regions, independent of the next group in row 3 and 4. If not set correctly, software might group regions that are not homologous to each other, leading to incorrect variant calls. Column 5 indicates whether a region is reverse complemented with respect to the other homologous region. A value of 1 denotes that the region is reverse complemented with respect to other regions in the same group. Row 4, column 5 is set to 1. This indicates that the region is homologous to the region in row 3 only if it is reverse complemented. <p>The DRAGEN installation package contains two VLRD bed files for hg19 and hs37d5 reference genomes under /opt/edico/examples/VLRD. You can use these bed files as is for running VLRD, or as an example to generate a custom bed file.</p>
--enable-vlrd-map-align-output	If set to true, VLRD outputs a remapped BAM/SAM file that only contains reads mapped to the regions that were processed by VLRD.

Unique Molecular Identifiers

DRAGEN can process data from whole genome and hybrid-capture assays with unique molecular identifiers (UMI). UMIs are molecular tags added to DNA fragments before amplification to determine the original input DNA molecule of the amplified fragments. UMIs help reduce errors and biases

introduced by DNA damage such as deamination before library prep, PCR error, or sequencing errors.

To use the UMI Pipeline, the input reads files must be from a paired-end run. Input can be pairs of FASTQ files or aligned/unaligned BAM input.

DRAGEN supports the following UMI types:

- Dual, nonrandom UMIs, such as TruSight Oncology (TSO) UMI Reagents or IDT xGen Prism.
- Dual, random UMIs, such as Agilent SureSelect XT HS2 molecular barcodes (MBC) or IDT xGen Duplex Seq Adapters.
- Single-ended, random UMIs, such as Agilent SureSelect XT HS molecular barcodes (MBC) or IDT xGen dual index UMI Adapters.

DRAGEN uses the UMI sequence to group the read pairs by their original input fragment and generates a consensus read pair for each such group, or family. The consensus reduces error rates to detect rare and low frequency somatic variants in DNA samples with high accuracy. DRAGEN generates a consensus as follows.

1. Aligns reads.
2. Groups reads into groups with matching UMI and pair alignments. These groups are referred to as families.
3. Generates a single consensus read pair for each read family.

These generated reads have higher quality scores than the input reads and reflect the increased confidence gained by combining multiple observations into each base call.

UMI workflow is only compatible with small variant calling and SV in DRAGEN.

UMI Input

Enter UMIs in one of the following formats:

- Read name—The UMI sequence is located in the eighth colon-delimited field of the read name (QNAME). For example, NDX550136:7:H2MTNBDXX:1:13302:3141:10799:AAGGATG+TCGGAGA
- BAM tag—The UMI is present as an RX tag in prealigned or aligned BAM file (standard SAM format).
- FASTQ file—The UMI is located in a third FASTQ file using the same read order as the read pairs.

To create FASTQ, append the UMI to the read name, and then specify the appropriate OverrideCycles setting in the DRAGEN BCL conversion tool (see [BCL Data Conversion on page 613](#)). DRAGEN supports UMIs with two parts each with a maximum of 8 bp and separated by +, or a single UMI with a maximum of 15 bp.

The UMI workflow must be executed using a set of reads that correspond to a unique set of RGSM/RGLB. DRAGEN supports multiple lanes if all lanes correspond to the same RGSM/RGLB set.

DRAGEN UMI does not support a tumor-normal analysis, because a tumor-normal run corresponds to two different RGSM. In a tumor-normal run, one sample name is used for tumor and one sample name is used for normal. DRAGEN UMI supports one sample in a run.

If using a BAM file or a list of FASTQ files as the input, the input might contain multiple samples. DRAGEN checks if only one sample is included in the run and if the sample uses only a single, unique RGLB library. DRAGEN also accepts a library that was spread across multiple lanes. If there is a single sample and single library, DRAGEN processes all included reads. If there are multiple samples or multiple libraries, DRAGEN aborts analysis with an error.

UMI Input Correction Table

For dual, nonrandom UMIs, you can provide a predefined UMI correction table or a list of valid UMI sequences as input. To create the UMI correction table, use a tab-delimited file, include a header, and add the following fields.

Field	Value
UMI	The UMI sequence. For example, ACGTAC.
IsValid	Specify if the UMI sequence is valid. Enter either: TRUE or FALSE.
NearestCodes	Colon-separated list of nearest UMI sequences. For example, ACGTAA:ACGTAT.
SecondNearestCodes	Colon-separated list of second nearest sequences. For example, ACGGAA:ACGGAT.

If customized correction table is not specified, DRAGEN uses the default table for TruSight Oncology (TSO) UMI Reagents located at `src/config/umi_correction_table.txt`. Alternatively, you can provide a file for whitelisted nonrandom UMI with valid UMI sequence one per line. DRAGEN then autogenerates a UMI correction table with hamming distance of one.

UMI Options

Option	Description
<code>--umi-library-type</code>	Set the batch option for different UMIs correction. Three batch modes are available that optimize collapsing configurations for different UMI types. Use one of the following modes: <ul style="list-style-type: none"> <i>random-duplex</i>—Dual, random UMIs. <i>random-simplex</i>—Single-ended, random UMIs. <i>nonrandom-duplex</i>—Dual, nonrandom UMIs. To use this option, provide either <code>--umi-nonrandom-whitelist</code> or <code>--umi-correction-table</code>.

Option	Description
--umi-min-supporting-reads	<p>Specify the number of matching UMI inputs reads required to generate a consensus read. Any family with insufficient supporting reads is discarded. For example, the following are the recommended settings for FFPE and ctDNA.</p> <ul style="list-style-type: none"> • [FFPE] If the variant > 1%, use --umi-min-supporting-reads=1 with the --vc-enable-umi-solid variant caller parameter. For more information on variant caller options, see Variant Caller Options on page 113. • ctDNA If the variant < 1%, use --umi-min-supporting-reads=2 with the --vc-enable-umi-liquid variant caller parameter. For more information on variant caller options, see Variant Caller Options on page 113
--umi-enable	To enable read collapsing, set the --umi-enable option to true. This option is not compatible with --enable-duplicate-marking because the UMI pipeline generates a consensus read from a set of candidate input reads, rather than choosing the best nonduplicate read. If using the --umi-library-type option, --umi-enable is not required.
--umi-emit-multiplicity	<p>Set the consensus sequence type to output. DRAGEN UMI allows you to collapse duplex sequences from the two strands of the original molecules. Duplex sequence is typically ~20–60% of total library, depending on library kit, input material, and sequencing depth. Enter one of the following consensus sequence types:</p> <ul style="list-style-type: none"> • <i>both</i>—Output both simplex and duplex sequences. This option is the default. • <i>simplex</i>—Output only simplex sequences. • <i>duplex</i>—Output only duplex sequences.
--umi-source	Specify the input type for the UMI sequence. The following are valid values: qname, bamtag, fastq. If using --umi-source=fastq, provide the UMI sequence from FASTQ file using --umi-fastq.
--umi-correction-table	Enter the path to a customized correction table. By default, Local Run Manager uses lookup correction with a built-in table for the Illumina TruSight Oncology and Illumina for IDT UMI Index Anchor kits.
--umi-nonrandom-whitelist	Enter the path for a customized, valid UMI sequence.
--umi-metrics-interval-file	Enter the path for target region in BED format.

Nonrandom and Random UMI Correction

DRAGEN processes UMIs by grouping reads by UMI and alignment position. If there are sequencing errors in the UMIs, DRAGEN can correct and detect small sequencing errors by using a lookup table or by using sequence similarity and read counts. You specify the type of correction with the `--umi-library-type` or `--umi-correction-scheme` option using the values `lookup`, `random`, or `none`.

For sparse sets of nonrandom UMIs, it is possible to create a lookup table that specifies which sequence can be corrected and how to correct it. This correct file scheme works best on UMI sets where sequences have a minimum hamming/edit distance between them. By default, DRAGEN uses lookup correction with a built-in correct table for the Illumina TruSight Oncology and Illumina for IDT UMI Index Anchor kits. Specify the path for your correction file using the `--umi-correction-table` option. If you are using a different set of nonrandom UMIs, contact Illumina Technical Support for information on generating the corresponding correction file.

In the random UMI correction scheme, DRAGEN must infer which UMIs at a given position are likely to be errors relative to other UMIs observed at the same position. The error modes include small UMI errors, such as one mismatch or UMI jumping or hopping artifact from library prep. DRAGEN accomplishes this as follows.

- Groups reads by fragment alignment position.
- Within a small fuzzy window at each position, groups the reads first by the exact UMI sequence, which forms a family.
- Estimate UMI jumping or hopping probability through insert size distribution and number of distinct UMI at certain positions.
- Within a fuzzy window, calculates pair-wise likelihood ratio to assess if two families with different UMI sequences and genomic positions are derived from same original molecule.
- Merges families with likelihood lower than threshold. The default threshold is 1.

Merge Duplex UMIs

Duplex UMI adapters simultaneously tag both strands of double-stranded DNA fragments. It is then possible to identify reads resulting from amplification of each strand of the original fragment.

DRAGEN considers two collapsed read pairs to be the sequence of two strands of the same original fragment of DNA if they have the same alignment position (within a fuzzy window), complementary orientations, and their UMIs are swapped from Read 1 and Read 2. If there is only single-ended UMI, DRAGEN compares the start-end position of families from two strands and computes pair-wise likelihood to determine if they are likely originated from two distinct families or should be merged as a duplex sequence. By default, DRAGEN outputs both simplex and duplex consensus sequences. To change the consensus sequence output type, use `--umi-emit-multiplicity`.

Example UMI Commands

Generate Consensus BAM from FASTQ

The following is an example DRAGEN command for generating a consensus BAM file from input reads with Illumina UMIs:

```
dragen \
-r <REF> \
-1 <FQ1> \
-2 <FQ2> \
--output-dir <OUTPUT> \
--output-file-prefix <PREFIX> \
--enable-map-align true \
--enable-sort true \
--umi-enable true \
--umi-correction-scheme=lookup \
--umi-min-supporting-reads 2
```

Use FASTQ UMI Input

To run with other random UMI library types, change `--umi-library-type` to `random-simplex` or `random-duplex`.

```
dragen \
-r <REF> \
-1 <FQ1> \
-2 <FQ3> \
--umi-source=fastq \
--umi-fastq <FQ2> \
--output-dir <OUTPUT> \
--output-file-prefix <
PREFIX> \
--enable-map-align true \
--enable-sort true \
--umi-library-type nonrandom-duplex \
--umi-metrics-interval-file [valid target BED file]
```

Use Customized Correction Table

```
dragen \
-r <REF> \
-1 <FQ1> \
```

```
-2 <FQ2> \
--umi-correction-table <valid umi correction table> \
--output-dir <OUTPUT> \
--output-file-prefix <PREFIX> \
--enable-map-align true \
--enable-sort true \
--umi-library-type nonrandom-duplex \
--umi-metrics-interval-file <valid target BED file>
```

UMI Outputs

Collapsed BAM

If you enable BAM output, DRAGEN generates a `<output_prefix>.bam` that includes all UMI consensus reads. The QNAMEs for the reads are generated based on the following convention.

`consensus_read_refID1_pos1_refID2_pos2_orientation`

- `refID1`—The reference ID of Read 1.
- `pos1`—The genomic position of Read 1.
- `refID2`—The reference ID of Read 2.
- `pos2`—The genomic position of Read 2.
- `orientation`—The orientation of Read 1 and Read 2. Orientation can be one of the following values. Position refers to the outermost aligned position of the read and is adjusted for soft clips.
 - 1—Read 1 is forward and Read 2 is reverse. The starting position for Read 1 is less than or equal to the Read 2 end position.
 - 2—Read 1 is reverse and Read 2 is forward. The starting position for Read 2 is greater than or equal to the Read 1 end position.
 - 3—Read 1 is forward and Read 2 is reverse. The starting position for Read 1 is greater than the Read 2 end position.
 - 4—Read 1 is reverse and Read 2 is forward. The starting position for Read 2 is greater than the Read 1 end position.
 - 5—Read 1 and Read 2 are forward.
 - 6—Read 1 and Read 2 are reverse.

`XV` and `XW` tags are added to consensus reads specifying number of supporting reads in a collapsed family. `XV` tag indicates the number of fragments and `XW` tag indicates the number of duplex fragments.

UMI Metrics

DRAGEN outputs a `<output_prefix>.umi_metrics.csv` file that describes the statistics for UMI collapsing. This file summarizes statistics on input reads, how they were grouped into families, how UMIs were corrected, and how families-generated consensus reads. The following metrics can be useful when tuning the pipeline for your application:

- Discarded families—Any families having fewer than `--umi-min-supporting-reads` input or having a different duplex/simplex status than specified by `--umi-emit-multiplicity` are discarded. These reads are logged as `Reads filtered out`. The families are logged as `Families discarded`.
- UMI correction—Families may be combined in various ways. The number of such corrections are reported as follows.
 - Families shifted—Families with fragment alignment coordinates up to the distance specified in the `umi-fuzzy-window-size` parameter. The default `umi-fuzzy-window-size` parameter is 3.
 - Families contextually corrected—Families with the same fragment alignment coordinates and compatible UMIs are merged.
 - Duplex families—Families with close alignment coordinates and complementary UMIs are merged.

When you specify a valid path for `--umi-metrics-interval-file`, DRAGEN outputs a separate set of on target UMI statistics that contains only families within the specified BED file.

If you need to analyze the extent to which the observed UMIs cover the full space of possible UMI sequences, the histogram of unique UMIs per fragment position metric may be helpful. It is a zero-based histogram, where the index indicates a count of unique UMIs at a particular fragment position and the value represents the number of positions with that count.

The following figures and table describe available UMI metrics.

Figure 14 Fig 1. Read pairs with duplex UMI

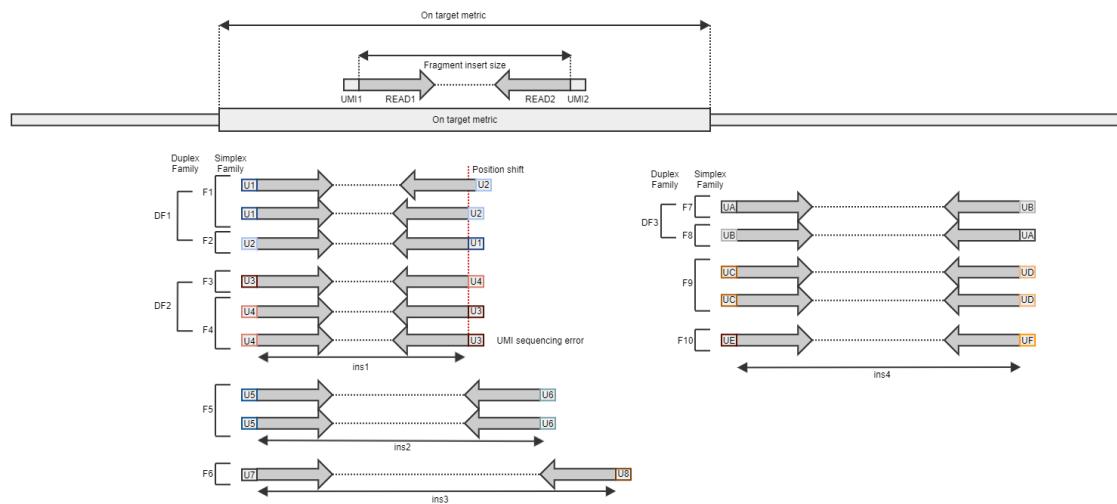


Figure 15 Fig 2. UMI error correction

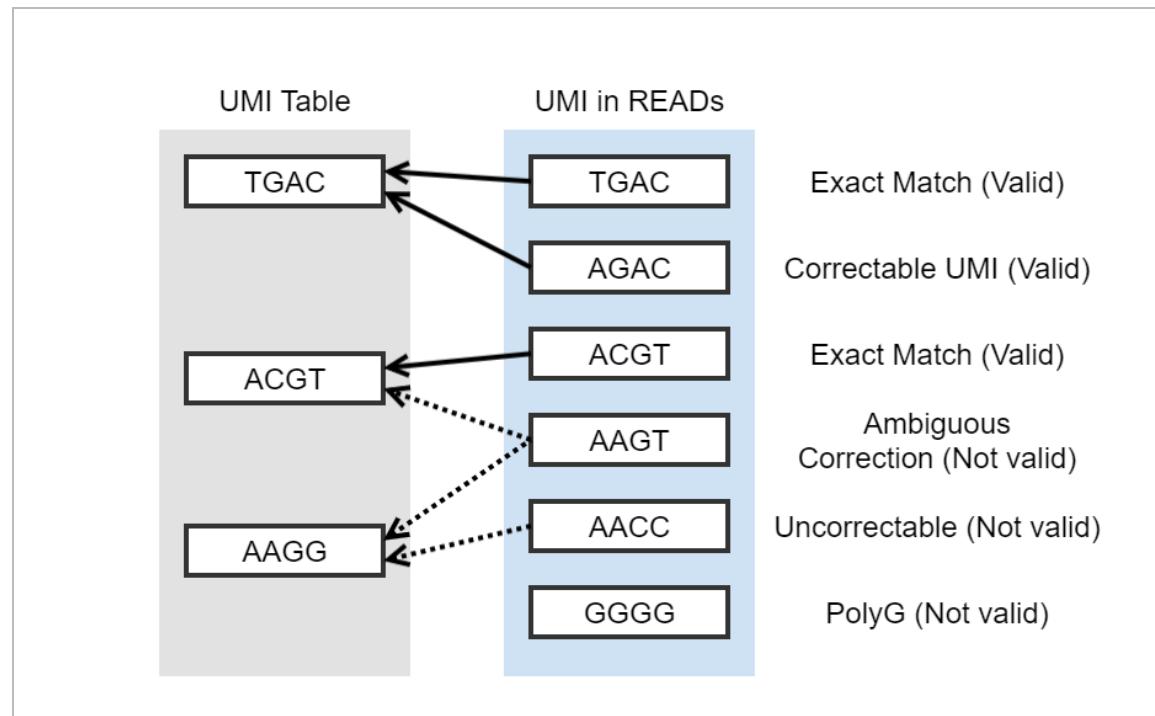
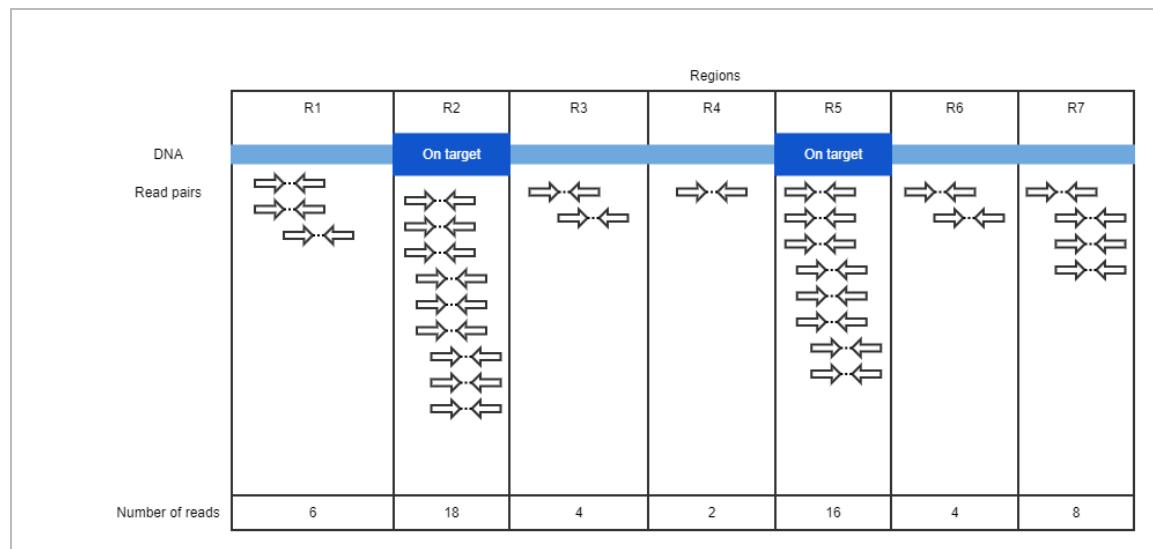


Figure 16 Fig 3. UMI collapsible regions



Metric	Description	Denominator of percentile	Example
Number of reads	Total number of reads.	N/A	Fig 1. Read pairs with duplex UMI: 14 pairs of read X 2 = 28 reads
Number of reads with valid or correctable UMIs	Number of reads for which the UMIs could be corrected based on the lookup table.	Number of reads	Fig 2. UMI error correction: Valid UMI read count (Exact match+Correctable UMI)
Number of reads in discarded families	Number of reads in discarded families. Families are discarded when there are not enough raw reads to support the family. (family size less than "--umi-min-supporting-reads"). For "--umi-emit-multiplicity=duplex" option, simplex families will be discarded.	Number of reads	Fig 1. Read pairs with duplex UMI: Number of reads in Families discarded (See "Families discarded" for more detail)
Reads with all-G UMIs filtered out	Number of reads filtered out due to all-G in UMI sequence.	Number of reads	Number of reads in discarded families + Reads with all-G UMIs + Number of unpaired reads

Metric	Description	Denominator of percentile	Example
Reads filtered out	Number of reads filtered out in total either for properties or in a discarded family.	Number of reads	Fig 2. UMI error correction: PolyG UMI read count
Reads with uncorrectable UMIs	Number of reads where the UMI could not be corrected.	Number of reads	Fig 2. UMI error correction: Uncorrectable + Ambiguous correction + PolyG
Total number of families	Number of simplex collapsed reads.	N/A	Read pairs with duplex UMI: F1~F10.
Families contextually corrected	Number of families that have some contextual correction. Contextual correction is based on other families at the same mapping location including UMI sequencing error and UMI jumping.	Total number of families	Fig 2. UMI error correction: Family count of correctable UMI
Families shifted	Number of families that have some shift correction. Shift correction merges families with fragment alignment coordinates up to the distance specified by the <code>umi-fuzzy-window-size</code> parameter.	Total number of families	Fig 1. Read pairs with duplex UMI: First read pair of DF1 (If shifted distance <= "umi-fuzzy-window-size")
Families discarded	Number of families filtered out by failing min supporting reads criteria or umi-emit type of simplex/duplex.	Total number of families	Fig 1. Read pairs with duplex UMI: Families discarded by min-support-reads + Families discarded by duplex/simplex (See below for detail)
Families discarded by min-support-reads	Number of families filtered out by failing min supporting reads criteria.	Total number of families	Fig 1. Read pairs with duplex UMI: Number of families size less than "umi-min-supporting-reads" option Size 1: F6, F10 Size 2: DF3, F5, F9 Size 3: DF1, DF2

Metric	Description	Denominator of percentile	Example
Families discarded by duplex/simplex	Number of families filtered out by failing umi-emit type of simplex/duplex.	Total number of families	Fig 1. Read pairs with duplex UMI: Number of simplex families (F5, F6, F9, F10) filtered. Note that simplex reads are only filtered if umi-emit-multiplicity=duplex (default: both)
Families with ambiguous correction	Number of families where the UMI cannot be corrected because more than one possible UMI corrections exists.	Total number of families	Fig 2. UMI error correction: Number of families of ambiguous correction UMI
Duplex families	Number of families that are merged as duplex (both strands).	Consensus pairs emitted	1. Read pairs with duplex UMI: DF1, DF2, DF3
Consensus pairs emitted	Number of collapsed reads in output BAM.	N/A	Fig 1. Read pairs with duplex UMI: Depends on umi-emit-multiplicity=simplex/duplex/both, umi-min-supporting-reads=x simplex=F1~F10 (F2, F3, F6, F7, F8, F10 filtered if $x \geq 2$) duplex=DF1, DF2, DF3 both=DF1, DF2, DF3, F5, F6, F9, F10 (F6, F10 filtered if $x \geq 2$)
Mean family depth	Average number of read pairs per family. Filtered reads and families are excluded.	N/A	Fig 1. Read pairs with duplex UMI: Number of reads per family: DF1=3, DF2=3, DF3=2, F5=2, F6=1, F9=2, F10=1 Mean family depth = $(3+3+2+2+1+2+1)/7 = 2$
Histogram of num supporting fragments	Number of families with zero raw reads, one raw read, two raw reads, three raw reads, etc.	N/A	Fig 1. Read pairs with duplex UMI: 0 reads: None 1 reads: F6, F10 = 2 (0 if umi-min-supporting-reads=2) 2 reads: DF3, F5, F9 = 3 3 reads: DF1, DF2 = 2 Histogram = {0 0 3 2}

Metric	Description	Denominator of percentile	Example
Number of collapsible regions	Number of regions.	N/A	Fig 3. UMI collapsible regions: R1~R7
Min collapsible region size (num reads)	Number of reads in the least populated region.	N/A	Fig 3. UMI collapsible regions: 2 reads (R4)
Max collapsible region size (num reads)	Number of reads in the most populated region.	N/A	Fig 3. UMI collapsible regions: 18 reads (R2)
Mean collapsible region size (num reads)	Average number of reads per region.	N/A	Fig 3. UMI collapsible regions: 8.3
Collapsible region size standard deviation	Standard deviation of the number of reads per region.	N/A	Fig 3. UMI collapsible regions: 5.8
On target number of reads	Number of reads that overlapped with UMI target interval --umi-metrics-interval-file.	N/A	Fig 1. Read pairs with duplex UMI, Fig 3. UMI collapsible regions: All On target metrics are same as corresponding metric but only considering fragments overlap with target intervals. i.e. DF3, F9, F10 in figure1 and R1, R3, R4, R6, R7 in figure3 are excluded from metric
On target number of reads with valid or correctable UMIs	Number of reads with a UMI that matched a UMI in the lookup table, including error allowance, and overlapped with UMI target interval.	On target number of families	
On target number of reads in discarded families	Number of reads in discarded families that overlapped with the UMI target interval.	On target number of families	

Metric	Description	Denominator of percentile	Example
On target duplex families	Number of families that are merged as duplex among all the families that are overlapped with UMI target interval.	On target number of families	
On target mean family depth	Average number of reads per family that overlapped with UMI target interval.	N/A	
On target families discarded	Number of families that overlapped with UMI target interval filtered out by failing min supporting reads criteria or umi-emit type of simplex/duplex.	On target number of families	
On target families discarded by min-support-reads	Number of families that overlapped with UMI target interval filtered out by failing min supporting reads criteria.	On target number of families	
On target families discarded by duplex/simplex	Number of families that overlapped with UMI target interval filtered out by failing umi-emit type of simplex/duplex.	On target number of families	
On target families with ambiguous correction	Number of families that overlapped with UMI target interval where the UMI cannot be corrected because more than one possible UMI corrections exists.	On target number of families	

Metric	Description	Denominator of percentile	Example
Histogram of unique UMIs per fragment position	Number of positions with zero UMI sequences, one UMI sequence, two UMI sequences, etc.	N/A	Fig 1. Read pairs with duplex UMI: 0 UMI sequence: None 1 UMI sequences: ins2 (F5), ins3 (F6) 2 UMI sequences: ins1 (DF1, DF2) 3 UMI sequences: ins4 (DF3, F9, F10) Histogram = {0 2 1 1}
Total families in probability model estimation	Total number of families used in estimation of UMI jumping rate and fragment size distribution used for probabilistic family merging.	N/A	
Number of potential Jumping Families	Total number of families that are potential UMI jumping candidates and the corresponding ratio.	Total Families in Probability Model Estimation	

Multicaller Workflows

DRAGEN supports running multiple tools in a single workflow.

The *enable-component* flag controls which components are enabled or disabled. DRAGEN constructs a workflow using the enabled components and automatically resolves any component inconsistencies. When possible, DRAGEN runs components in parallel.

Each component has a set of options that configures input settings, internal algorithm parameters, or output files and filtering criteria. Refer to the individual component sections for more details.

As an example, a different BED file may be provided separately for each caller:

- cnv-target-bed
- sv-call-regions-bed
- vc-target-bed

Some options, such as *output-directory* and *sample-sex*, are shared amongst callers.

Each variant caller produces its own set of VCFs and metric output files.

Example Component Commands

```
enable-map-align
enable-sort
enable-duplicate-marking
enable-variant-caller
enable-cnv
enable-sv
```

Input Formats

DRAGEN accepts the following common standard NGS input formats:

- FASTQ (`fastq-file1` and `fastq-file2`)
- FASTQ List (`fastq-list`)
- BAM (`bam-input`)
- CRAM (`cram-input`)

Somatic workflows can use tumor equivalent input files (eg, `tumor-bam-input`).

When running from unaligned reads, the reads first go through the map/align component to produce alignments which continue downstream to the variant callers. When running from prealigned reads, DRAGEN supports re-aligning with the map/align component or using the existing alignments from the source input.

Multicaller Command Line Example

The following example demonstrates best practices for combining command line options from single caller scenarios to create a multicaller workflow. The example consists of the following steps:

- Configure the INPUT options.
- Configure the OUTPUT options.
- Configure MAP/ALIGN depending on if realignment is desired or not.
- Configure the VARIANT CALLERs based on the application.
- Build up the necessary options for each component separately, to enable reuse in the final command line.

The following are partial templates that can be used as starting points. Adjust them accordingly for your specific use case.

```
#!/bin/bash
set -euo pipefail
```

```

DRAGEN_HASH_TABLE=<REF_DIR>
FASTQ1=<fastq1>
FASTQ2=<fastq2>
RGSM=<RGSM>
RGID=<RGID>
OUTPUT=<OUT_DIR>
PREFIX=<OUT_PREFIX>

INPUT_OPTIONS=""
--ref-dir $DRAGEN_HASH_TABLE \
--fastq-file1 $FASTQ1 \
--fastq-file2 $FASTQ2 \
--RGSM $RGSM \
--RGID $RGID \
"

OUTPUT_OPTIONS=""
--output-directory $OUTPUT \
--output-file-prefix $PREFIX \
"

MA_OPTIONS=""
--enable-map-align true \
... <any other optional settings> \
"

CNV_OPTIONS=""
--enable-cnv true \
... <any other optional settings> \
"

SNV_OPTIONS=""
--enable-variant-caller true \
... <any other optional settings> \
"

SV_OPTIONS=""
--enable-sv true \
... <any other optional settings> \
"

CMD="
dragen \
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \
"

```

```
$MA_OPTIONS \
$CNV_OPTIONS \
$SNV_OPTIONS \
$SV_OPTIONS \
"
echo $CMD
bash -c $CMD
```

Germline

The following table summarizes the support for some input formats and variant callers. Some supported features and callers are not listed in the table.

GERMLINE	FASTQ with Map/Align	BAM/CRAM	BAM/CRAM with Map/Align
CNV + SNV	Supported	Supported	Supported
CNV + SV	Supported	Supported	Supported
SNV + SV	Supported	Supported	Supported
CNV + SNV + SV	Supported	Supported	Supported

Somatic

Somatic workflows specify both tumor and normal inputs. The need for potentially two input files (tumor and matched normal) as well as the need for a matched normal SNV VCF for the Somatic CNV caller means extra care has to be taken.

One recommended tumor/normal workflow first starts with running matched normal through Germline Workflow.

1. Run matched normal through Germline workflow (CNV + SNV + SV + ...). The workflow generates the matched normal SNV VCF.
2. Run tumor and matched normal through Somatic workflow (CNV + SNV + SV + ...)

Optionally, a full tumor/normal analysis can be done in a single execution if both the SNV and CNV modules are enabled, by leveraging the BAF information directly from the small variant caller. See the Somatic CNV section for more details. In brief, this requires the use of `--enable-variant-caller true` and `--cnv-use-somatic-vc-baf true`.

```
#!/bin/bash
set -euo pipefail

DRAGEN_HASH_TABLE=<REF_DIR>
```

```
TUMOR_BAM=<TUMOR_BAM>
NORMAL_BAM=<NORMAL_BAM>
OUTPUT=<OUT_DIR>
PREFIX=<OUT_PREFIX>

INPUT_OPTIONS=""
--ref-dir $DRAGEN_HASH_TABLE \
--tumor-bam-input $TUMOR_BAM \
--bam-input $NORMAL_BAM \
"

OUTPUT_OPTIONS=""
--output-directory $OUTPUT \
--output-file-prefix $PREFIX \
"

MA_OPTIONS=""
--enable-map-align false \
... <any other optional settings> \
"

CNV_OPTIONS=""
--enable-cnv true \
... <any other optional settings> \
"

SNV_OPTIONS=""
--enable-variant-caller true \
... <any other optional settings> \
"

SV_OPTIONS=""
--enable-sv true \
... <any other optional settings> \
"

CMD="
dragen \
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \
$MA_OPTIONS \
$CNV_OPTIONS \
$SNV_OPTIONS \
$SV_OPTIONS \
"
```

```
echo $CMD
bash -c $CMD
```

The following table lists the various combinations that are supported under the tumor/normal mode of operation.

Tumor normal	FASTQ with Map/Align	BAM/CRAM	BAM/CRAM with Map/Align
CNV + SNV	Supported	Supported	Not Supported
CNV + SV	Supported	Supported	Not Supported
SNV + SV	Supported	Supported	Not Supported
CNV + SNV + SV	Supported	Supported	Not Supported

To run in tumor only mode, remove the matched normal input from the INPUT options and configure each individual caller to run in tumor only mode. The following table lists the combinations that are supported under the tumor only mode of operation.

Tumor Only	FASTQ with Map/Align	BAM/CRAM	BAM/CRAM with Map/Align
CNV + SNV	Supported	Supported	Supported
CNV + SV	Supported	Supported	Supported
SNV + SV	Supported	Supported	Supported
CNV + SNV + SV	Supported	Supported	Supported

WES analysis is supported if the mode is supported in singe caller mode and there is no input configuration conflict.

For WES analysis, CNV requires a panel of normals regardless of whether it is Tumor Normal or Tumor Only analysis.

Indel Re-alignment (Beta)

Overview

The DRAGEN Indel Re-aligner is a consensus based re-alignment step, independent from other DRAGEN callers and pipelines. Re-aligned reads are reflected in the output BAM file, and their original alignment is described in an OA tag. The implementation is similar to the Indel Re-aligner tool that was found in GATK3. The tool is designed to reduce false positive SNPs by considering evidence of near-by indels.

Description

The pipeline is comprised of two concurrent steps: Interval creation and re-alignment. The interval creation step identifies genomic intervals for which there is evidence of insertions or deletions in the CIGAR's of properly paired (if paired) reads aligned with positive mapq. To output these intervals as a text file, use the command line argument `--ir-write-intervals-file=true`. Each line will describe a genomic interval as `chrom:start-end`, or `chrom:start` for intervals of length one. The start and end positions are both inclusive and 1-based. The intervals file will be written to the DRAGEN output directory, with the suffix `realign-intervals.txt`.

For each genomic interval, the realignment step groups all aligned reads that intersect the interval. If there are more than `ir-max-num-reads` reads that intersect the interval, it is skipped. The following reads are then discarded from the re-alignment analysis:

- Non-primary aligned reads.
- Reads whose mapping quality is zero.
- Paired end reads that mapped to different contigs.
- Paired end reads that mapped to the same contig with start positions more than `ir-max-distance-between-mates` apart.

Reads that have not been skipped are candidates for re-alignment. If there are more than `ir-max-num-candidates` candidates, the interval is skipped. From each re-alignment candidate, a consensus read is generated from any read that has a single indel that is not the first or last CIGAR operation excluding clip operations. If there are more than `ir-max-number-consensus` consensus reads, the interval is skipped. Each re-alignment candidate is then scored against each consensus to determine the winning consensus. If the combined score for the interval against the winning consensus is better than the score against the reference by a difference of at least `ir-realignment-threshold`, the reads start position, CIGAR, and NM tag are updated to reflect the re-alignment. The scoring used is hamming distance weighted by base qualities. OA tags that describe the original alignment are added to any re-aligned reads. Mate positions of reads whose mate was re-aligned are updated as well.

When the re-alignment step is complete, a summary will be printed to standard out. It will describe the number of intervals found, sum of the lengths of all intervals, number of reads that intersected intervals, number of reads that got re-aligned, and the number of reads that were skipped due to memory constraints. Such reads will be documented in the DRAGEN log. This may happen in regions with very deep coverage.

Limitations

The DRAGEN Indel Re-aligner is designed to improve the quality of the DRAGEN BAM output for downstream analysis. The DRAGEN small variant caller pipeline does not read the output BAM, and has its own internal haplotype assembly step which usually recovers most of the artifacts found during Indel Re-alignment. Limited testing has shown that there may be a small improvement in DRAGEN small

variant calls when Indel Re-alignment is enabled. However, Indel Re-alignment will slow down a DRAGEN Map/Align + VC run roughly by a factor of two. For that reason, it is not recommended to enable Indel Re-alignment with the DRAGEN VC, and it is not enabled by default.

The Indel Re-alignment pipeline cannot run with:

- The UMI pipeline
- The Methylation pipelines.
- `--qc-coverage-ignore-overlaps=true`
- SA tag generation (`--generate-sa-tags=true`)
- The Expansion Hunter pipeline.

Command Line Arguments

Name	Description	Default Value
<code>enable-indel-realigner</code>	Enable indel re-alignment	False
<code>ir-write-intervals-file</code>	Output a file with the reference intervals that contain evidence for re-alignment.	False
<code>ir-max-num-reads</code>	Max number of reads in an interval for re-alignment.	20,000
<code>ir-max-num-candidates</code>	Max number of re-alignment candidates in an interval for re-alignment.	256
<code>ir-max-num-consensus</code>	Max number of consensus reads in an interval for re-alignment.	256
<code>ir-max-distance-between-mates</code>	Max number of re-alignment candidates in an interval for re-alignment.	100,000
<code>ir-realignment-threshold</code>	Minimal improvement of sum of mismatching base qualities to merit realignment.	50

Star Allele Caller

The Star Allele Caller identifies the genotypes and metabolism status of the following PGx genes. They are included in [FDA's PGx recommendations](#) or have [CPIC Level A designation](#): CACNA1S, CFTR, CYP2C19, CYP2C9, CYP3A5, CYP4F2, IFNL3, RYR1, NUDT15, SLCO1B1, TPMT, UGT1A1, VKORC1, DPYD, G6PD, MT-RNR1, BCHE, ABCG2, NAT2, F5, and UGT2B17. Star Allele Caller finds optimal genotypes for the genes based on Star Allele definitions from the following resources. It calls metabolism status based on a PharmCAT resource file that provides mappings between genotypes and

phenotypes. The primary support for the Star Allele Caller is for human reference hg38 for which it supports the above mentioned genes. In addition, Star Allele Caller supports the following genes on references hg19 and GRCh37 : CACNA1S, CYP2C19, CYP2C9, CYP3A5, CYP4F2, IFNL3, NUDT15, SLCO1B1, VKORC1, DPYD, ABCG2, F5.

Star allele definition resources for hg38

For genes CACNA1S, CFTR, CYP2C19, CYP2C9, CYP3A5, CYP4F2, IFNL3, RYR1, NUDT15, SLCO1B1, TPMT, UGT1A1, VKORC1, DPYD, G6PD, MT-RNR1, ABCG2 the allele definitions are sourced from PharmGKB. Because BCHE does not have defined star alleles, the Star Allele Caller checks if a sample is positive for any of the variants that are reported.

Star allele definition resources for hg19/GRCh37

For genes CYP2C19, CYP2C9, CYP3A5, CYP4F2, NUDT15, SLCO1B1, DPYD, the definitions are sourced from PharmVAR. For the remaining hg19/GRCh37 genes, ie, ABCG2, CACNA1S, IFNL3, F5 and VKORC1 - the allele definitions have been lifted from their corresponding definitions for hg38, which are sourced from PharmGKB.

Functionality

The Star Allele Caller has the following features.

- It calls star allele genotypes from different types of genomic data like FASTQ, BAM, gVCF, VCF
- It provides additional details about the genotype call, including a confidence score
- It assumes genotypes for missing positions to be ref - these positions are listed in the output
- It assumes filtered genotype calls to be ref - these records are also listed in the output
- If multiple optimal diplotypes are satisfied, then it lists them all
- It supports different versions of the human reference hg38
- For the genes UGT2B17 and CYP2C19, the caller analyzes CNV calls to detect star alleles.

Input files and command line options

The Star Allele Caller can accept as input, different forms of sequence data such as FASTQs files, BAM/CRAM files or gVCF/VCF files. In the simplest case, the caller takes a DRAGEN gVCF file as input. The following is an example of the command line for this use case.

```
dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/9 \
--star-allele-gvcf /staging/test/data/NA12878.gvcf \
--star-allele-cnv-vcf /staging/test/data/NA12878.cnv.vcf.gz \
--output-directory /staging/test/output \
```

```
--output-file-prefix NA12878_dragen \
--enable-star-allele true
```

Contrary to a variant-only VCF file, a DRAGEN gVCF file contains the genotypes for all positions in a genome. Although the gVCF format is the preferred format for the caller, it can also accept a standard variant-only VCF file as input. The command line for that case will be the same as above, with the VCF file passed instead of a gVCF file. Also, the CNV-VCF file is optional - in this case the Star Allele Caller will not call star alleles that are detected through CNV analysis.

The following is an example of this use case, with only a variant only VCF file as input.

```
dragen \
-r /staging/human/reference/hg38_alt_aware+cnv+hla+rna_v2/DRAGEN/9 \
--star-allele-gvcf /staging/test/data/NA12878.vcf \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--enable-star-allele true
```

The small variant VCF/gVCF and CNV-VCF files should meet the following specifications.

- Must be aligned to the same human reference that is passed through the `-r` option.
- Variants should follow a parsimonious left aligned variant representation format.
- Complex variants - for example, representing closely located, independent variants, in a single record - are NOT supported.

VCF/gVCF files can be substituted with a compressed GZ file (ie `<file_name>.vcf.gz` or `<file_name>.gvcf.gz`). If a BAM or FASTQ file is passed as input, then the preferred command to use for Star Allele Caller is the option `--enable-pgx` which turns on all necessary components that the Star Allele Caller needs. The human reference needs to be passed as a command line option. The Star Allele Caller detects the reference version (ie, hg19, GRCh37 or hg38) and accordingly reads in the correct allele definitions.

```
dragen \
-r /staging/human/reference/hg38_alt_aware+cnv+hla+rna_v2/DRAGEN/9 \
--bam-input /staging/test/data/NA12878.bam \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--enable-map-align false \
--enable-pgx true
```

Note that the directory passed with the `-r` option points to the directory containing the DRAGEN hash table for a reference, not the raw reference FASTA. DRAGEN team can provide these hash tables or they may be built using the following command (where `hg38.fa` is a reference FASTA).

```
dragen \
--build-hash-table true \
--ht-reference hg38.fa \
--output-directory <directory_for_ref_hashtables>
```

Once the hash tables are generated, the path <directory_for_ref_hashtables> can then be passed with the `-r` parameter. If the BAM file is not aligned to the available hg38 reference, automatic remapping can be done by setting the option `--enable-map-align true`. If the user has a BAM file that is pre-aligned to a custom hg38 reference, and wishes to use this reference for the Star Allele Caller, then the user needs to create a DRAGEN reference hash table, which can be built using the above command.

For passing a FASTQ file as input, additional options, `--RGID` and `--RGSM` need to be set in the command line. An example of the command line for this use case as follows.

```
dragen \
-r /staging/human/reference/hg38_alt_aware+cnv+hla+rna_v2/DRAGEN/9 \
-1 /scratch/NA11829.fq1.gz \
-2 /scratch/NA11829.fq2.gz \
--RGID DRAGEN_RGID \
--RGSM DRAGEN_RGSM \
--enable-map-align true \
--output-directory /staging/test/output \
--output-file-prefix NA11829 \
--enable-pgx true
```

The setting `--enable-pgx` turns on other PGx callers such as CYP2D6, CYP2B6, and HLA - in addition to the Star Allele Caller. Note that to run the HLA caller, the passed reference must contain anchored_hla, a specific subdirectory with HLA-specific reference files. If a user wants to ONLY turn on the Star Allele Caller and not the other PGx callers, then in addition to enabling the Star Allele Caller, the variant caller also needs to be enabled. Optionally, the CNV caller should also be preferably enabled for analyzing CNV star alleles. An example of the command line for this use case is as follows.

```
dragen \
-r /staging/human/reference/hg38_alt_aware/DRAGEN/9 \
--bam-input /staging/test/data/NA12878.bam \
--output-directory /staging/test/output \
--output-file-prefix NA12878_dragen \
--enable-map-align false \
--enable-star-allele true \
--enable-variant-caller true \
--vc-emit-ref-confidence gvcf \
```

```
--enable-cnv true \
--cnv-enable-self-normalization true
```

Additional options with the variant caller such as `--vc-forcegt-vcf` should not be used when the Star Allele Caller is enabled.

Output files

Following completion of the DRAGEN Star Allele Caller run, the following four output files are produced.

- When the Star Allele Caller is run along with other callers, then the main output file, `<prefix>.targeted.json` contains the complete and detailed results for all genes. (When the caller is run in stand-alone mode then this file is not produced - check #2 for the relevant output for that case). This is an example output for one gene `DYPD` and for one sample `NA19374`.

```
{
  "dragenversion": "4.2.0-724-gb600fcf",
  "sample": "NA19374",
  "pharmcatMetabolismStatusResourceUrl": "https://github.com/PharmGKB/PharmCAT/blob/aeecfe5f787e95dfb31ede62884e287affef45b3/src/main/resources/org/pharmgkb/pharmcat/definition/gene_phenotypes.json",
  "star_allele": {
    "calls": [
      {
        "gene": "DYPD",
        "lastUpdate": "10/13/2021",
        "alleleDefinitionsUrl": "https://www.pharmgkb.org/page/dpydRefMaterials",
        "genotype": "./.",
        "pharmcatDescription": null,
        "pharmcatMetabolismStatus": null,
        "variants": "chr1:97515839:T:C,<NON_REF>:0/1:49:70:PASS;chr1:97883329:A:G,<NON_REF>:1/1:65:68:PASS;chr1:97573881:C:T,<NON_REF>:0/1:50:73:PASS",
        "variantStarAllelesFound": "c.1218G>A:c.1627A>G(*5):c.85T>C(*9A)",
      }
    ]
  }
}
```

```

        "minGQ": "49",
        "missingGenotypes": "",
        "filteredGenotypes": ""

    }
]

}

```

The fields in the json file are as follows.

- "dragenVersion": Version of DRAGEN that is being executed
- "sample": Sample name
- "pharmcatMetabolismStatusResourceUrl": Web URL for the PharmCAT resource used for calling metabolism status
- "gene": Gene name
- "lastUpdate": Last update for allele definitions for the gene
- "alleleDefinitionsUrl": Web URL for the star allele definitions file
- "genotype": Detected optimal genotype for the gene
- "pharmcatDescription": A description of the called genotype from PharmCAT resource file
- "pharmcatMetabolismStatus": Metabolism status for called genotype from PharmCAT resource file
- "variants": List of relevant variants for the gene
- "variantStarAllelesFound": List of satisfied star allele haplotypes corresponding to variants
- "minGQ": A confidence score for the genotype call
- "missingGenotypes": List of relevant positions for which GT is missing in gVCF input
- "filteredGenotypes": List of relevant positions for which GT is filtered in gVCF input

i | The latest version of the resource file from PharmCAT no longer carries the `pharmcatDescription` field. As a result, the field is deprecated in DRAGEN v4.2.

Each Star allele genotype contains one or two haplotypes (a haplotype for chrM gene MT-RNR1 and chrX gene G6PD for male samples, and a diplotype for all other genes) separated by a slash (eg `*1/*2`). Each haplotype is a pre-defined star allele and the definitions can be found under the allele definitions URL. Note that there may be some variance in star allele definitions and notations based on the resource and when it was last updated. The Star Allele Caller follows the PharmGKB

- definitions and notations exactly. When the Star Allele Caller cannot identify an optimal genotype for a gene, a no-call (./. or .) is made. In certain cases, more than one genotype is optimally satisfied, in that case all satisfied genotypes are listed, separated by a colon (eg *1/*2:*3/*4)
2. TSV and json files are produced when the Star Allele Caller is run stand-alone from a gVCF or vcf file or if the option --targeted-enable-legacy-output is set. This produces a <prefix>.star_allele.tsv which contains summarized star allele calls for each gene. This is an example for one gene from the TSV output. The fields are gene name and genotype.

```
UGT1A1 *36/*80+*37
```

- Additionally, a json file, <prefix>.star_allele.json is also produced when this option is set that contains the genotype calls for just the Star Allele Caller (contrary to the main output file, <prefix>.targeted.json that aggregates calls from all targeted callers). The format of this json file is the same as the json format described above.
3. A gVCF file, <prefix>.select.gvcf contains specific gVCF records that were used to make the star allele genotype calls. This file follows the specifications of a VCF file and is a subset of the VCF/gVCF file that is supplied as input (or generated by DRAGEN VC from the FASTQs/BAM/CRAM files).

High Coverage Analysis

While DRAGEN is capable of supporting up to 1000x coverage, its default settings are tuned for a more typical sample size in the ~100x range. If the processing of your large sample doesn't complete, or it gives unexpected results, there are options available to improve the behavior.

Users may want to analyze high amounts of data using the DRAGEN platform. In somatic contexts, it can be beneficial to sequence the tumor at a very high depth to detect mutations at even lower frequencies. DRAGEN reliably supports a total average coverage of up to 1000x. As the input read data can grow excessively, but the system memory is limited, DRAGEN can only keep a subset of the input in RAM at the same time. The area reserved for the read data is called bin_memory. Higher bin_memory size means that bigger chunks can be processed simultaneously, but less memory is available to the rest of DRAGEN or for other processes.

After the map-align step, reads are loaded into the bin_memory, looking for regions of zero coverage. A set of reads that spans two such zero-coverage loci is a callable region. The memory used by a callable region is determined by the number of reads and their length. For instance, a long region with few reads per position uses the same amount of memory as a short region with a spike in coverage. The size of a callable region must stay well below the size of the bin_memory. To this end, any callable region that surpasses the --vc-max-callable-region-memory-usage threshold is cut into smaller regions. Due to these cuts, the accuracy of variant calls in the vicinity may be affected.

Command Line Options

The following options can be used to change the bin_memory and callable region size.

--bin_memory	Set the amount of memory reserved for read data. Defaults to at least 20GB for germline and 40GB for a somatic run.
--vc-max-callable-region-memory-usage	Set the maximum size of a single callable region. Default is 13GB.

CheckFingerprint Software Design

Constraints, Assumptions and Risks

CheckFingerprint is broadly based on Picard CheckFingerprint. CheckFingerprint will output LOD score to indicate whether all the genetic data between two files from the same individual or not.

If LOD score is positive, those two samples come from the same individual. Otherwise, those two samples come from different individuals.

In general, the sign of LOD in summary file should be consistent with Picard CheckFingerprint summary file, but the exact values may be different.

Validation were done on whole-genome sequencing (WGS) data, mixing WGS samples and whole exon sequencing data.

Hardware Design

The CheckFingerprint module is designed to be run on a DRAGEN server to take advantage of FPGA acceleration during the analysis.

Software Design

Architecture Overview

The CheckFingerprint module has two comparison modes: Read and VCF

Read Mode

1. Takes input DNA data (FASTQ, BAM, cram) and loads the configuration files
2. Processes the read through DRAGEN (mapping, sorting, marking duplicates)
3. Calculates CheckFingerprint output by examining each read

```
./bin/dragen -r ../../dragen_hashtable/ -b [bam] --output-directory
[output_dir] \
--output-file-prefix [output_prefix] --enable-checkfingerprint true --
checkfingerprint-expected-vcf [input_expected_vcf]
```

VCF Mode

1. Takes input DNA data (FASTQ, BAM, cram) and loads the configuration files
2. Processes the read through DRAGEN and conducts variant calling
3. Calculates CheckFingerprint output by analyzing each variant site

```
./bin/dragen -r ../../dragen_hashtable/ -b [bam] --output-directory
[output_dir] \
--output-file-prefix [output_prefix] --enable-checkfingerprint true --
checkfingerprint-expected-vcf [input_expected_vcf] \
--checkfingerprint-enable-vcf-comparison true --enable-variant-caller true
```

Architecture Use Cases

The CheckFingerprint module is designed to check all the genomic data in input sample and expected samples coming from the same individual. The input sample can be in the format of FASTQs, BAM or CRAM files. The expected sample should be in VCF format. Users are required to provide the VCF file of expected samples `--checkfingerprint-expected-vcf [adr/to/vcf]`. To run the CheckFingerprint function, users must provide these input files through a command-line interface along with the option to enable the CheckFingerprint module `--enable-checkfingerprint true`. The reference genome is also a required input on the command line and must be a human reference genome build based on hg19, GRCh37 or hg38. Configuration files required to run the CheckFingerprint function are automatically loaded from the DRAGEN installation config directory based on the provided reference genome. To run VCF mode, users also need to enable the CheckFingerprint VCF mode and DRAGEN variant calling `--checkfingerprint-enable-vcf-comparison true --enable-variant-caller true`.

DRAGEN CheckFingerprint function produces two outputs: `.CheckFingerprint.detail.txt` and `.CheckFingerprint.summary.txt`.

`.CheckFingerprint.detail.txt` outputs LOD value for each SNP position in `haplotype_map` covered by expected vcf file

For example:

```
READ_GROUP EXPECTED_SAMPLE SNP SNP_ALLELES CHROM POSITION EXPECTED_GENOTYPE
OBSERVED_GENOTYPE LOD OBS_A OBS_B
IGNORE hg002 chr1:274 AG 1 908025 AG AA 0.0799204 0 0
IGNORE hg002 chr1:308 GA 1 916119 GG GG 0.350172 0 0
IGNORE hg002 chr1:473 CT 1 984039 CT CC 0.39524 0 0
```

For `.CheckFingerprint.summary.txt`, it outputs the LOD value between input and expected samples.

For example:

```

READ_GROUP EXPECTED_SAMPLE LL_EXPECTED_SAMPLE LL_RANDOM_SAMPLE LOD_EXPECTED_
SAMPLE HAPLOTYPES_WITH_GENOTYPES HAPLOTYPES_CONFIDENTLY_CHECKED HAPLOTYPES_
CONFIDENTLY_MATCHING HET_AS_HOM HOM_AS_HET HOM_AS_OTHER_HOM
IGNORE hg002 -568.153 -105.25 -462.903 12558 122 49 51 15 7

```

Human Interface Design

The user interface to the CheckFingerprint is the DRAGEN command line. The user will provide input read data along with a reference genome and instruct the DRAGEN binary to run the CheckFingerprint using the command line option `--enable-checkfingerprint true`. Information related to the processing of read data and analysis of CheckFingerprint will be reported to standard output. Two output files can be inspected to get the LOD values between input and expected samples.

Input

The input files used by DRAGEN CheckFingerprint are: FASTQ/BAM/CRAM (user input), haplotype map (configuration files), VCF file (user input).

The FASTQ/BAM/CRAM file should be whole genome sequencing data or whole exon sequencing data.

1. Haplotype Map

Haplotype map is a set of SNPs grouped into haplotype blocks (also known as linkage disequilibrium blocks). SNPs in haplotype map is used as fingerprinting.

`NAME` is a SNP identifier.

`MAF` is the minor allele frequency.

`ANCHOR_SNP` in haplotype map refers to the `NAME` of a SNP. SNPs with the same `ANCHOR_SNP` have high linkage disequilibrium with each other.

2. Expected VCF File

The VCF file of expected samples. It can contain multiple samples. CheckFingerprint compares input sample with all the samples in the expected VCF file, and calculates LOD values of each pair.

Output

DRAGEN CheckFingerprint function produces the following outputs:

- `.CheckFingerprint.detail.txt`- Outputs the LOD value for each SNP position in the haplotype_map.
- `.CheckFingerprint.summary.txt`- Outputs the LOD value between the input and expected samples.

Process Flow

The configuration and sequencing read data flow through the DRAGEN software as outlined below. Aligned reads come from prealigned BAM/CRAM input or reads aligned by the DRAGEN mapper component. All data flows through the system in memory until the generation of the output txt files. There are no intermediate output files.

Configuration

The configuration files were described previously and are required input for DRAGEN CheckFingerprint. As a result, only human reference genomes based on hg19, GRCh37 and hg38 are supported. The configuration files are stored in the config installation directory located at /opt/edico/config/

Installation

The CheckFingerprint function and dependent configuration files are installed along with the main DRAGEN installer (platform-specific). Reference genome files may need to be installed separately. After the installation, the main DRAGEN executable is located at /opt/bin/dragen.

Open Source Software (OSS)

OSS will not be applicable other than what is already in use by DRAGEN.

Compatibility

The CheckFingerprint function is available starting with DRAGEN v4.2.

Detailed Design

CheckFingerprint calculates the LOD score to identify whether two samples are from the same individual or not. A positive value indicates those two samples are from the same individual. A negative value indicates two samples are not match. LOD is in logarithmic scale (base 10). A LOD of 4 indicates it is 10,000 more likely that data matches the genotypes than not. A score that is close to 0 is inconclusive and can result from low coverage or missing informative genotypes. The identity check takes advantage of haplotype blocks defined in configuration file (hg38_nochr.map,hg19_nochr.map). It can improve statistic power for identity detection by checking SNPs in haplotype blocks.

In VCF mode, CheckFingerprint uses PL to estimate genotype probabilities.

If the expected sample VCF file is not provided, CheckFingerprint will automatically compare with default subset hg002 VCF file in CheckFingerprint configuration files.

The default output has no biological significances. It is used to provide an example output while missing expected samples.

Fault Handling

Command line parameters are validated and a non-zero exit code is returned from the DRAGEN command line if there are any errors in the provided options (eg, missing input file) along with a clear error message that can be corrected by the user. For identifiable errors during the analysis resulting from invalid input or bugs leading to an invalid internal state, the analysis is abruptly terminated with an error message. Depending on the nature of the error, the error message may not be actionable by the user in which case the user would need to provide the details of the error to the development team for troubleshooting. Errors may also arise that are not identified during the analysis. These errors can include hardware problems, invalid memory accesses, deadlock and other programming errors resulting in undefined behavior. In these cases the DRAGEN process will terminate and diagnostics information about the state of the program prior to the error will be available for troubleshooting by the development team.

Population Haplotyping (Beta)

The Haplotyping tool is a beta tool introduced in the DRAGEN Bio-It Platform v4.2. It supports the estimation of haplotypes from a population scale dataset using the packaging of the SHAPEIT5 Software (2022, Hofmeister RJ, Ribeiro DM, Rubinacci S., Delaneau O). It is designed to phase common variants as well as rare variants in a step-by-step mode. The following step-by-step workflow must be reproduced to phase each chromosome of the studied genome.

1. Phase Common step to estimate the haplotypes of common variants (variants with allele frequency above a given allele frequency threshold) on defined regions.
2. Common Ligate step to ligate the phased common variants from the previous step into a single chromosome.
3. Phase Rare step to add the haplotypes of rare variants (variants with allele frequency below a given allele frequency threshold) on defined regions to the common variant scaffold obtained in the previous step.
4. Concat All step to concatenate the haplotype regions obtained in the previous step into a single chromosome.

This tool provides best accuracy on population scale dataset with thousands of samples. It is intended to be run on multiple nodes to parallelize processes. A common use case of the Population Haplotyping tool is the generation of a custom reference panel to be used for the VCF Imputation pipeline.

The tool supports autosomes and mixed ploidy chromosomes for diploid species only. It does not use the FPGA accelerated capability and it can run on generic software only compute node.

 | The Population Haplotyping tool only supports input msVCF produced with the DRAGEN gVCF Genotyper tool.

Command Line Examples

The following is an example of required command to generate haplotypes on common and rare variants (with default allele frequency threshold) on population scale dataset:

1. Phase Common

```
dragen \
--enable-population-haplotype true \
--enable-phase-common true \
--ph-phase-common-input-list <path_to_txt_file> \
--ph-phase-common-input-region <string> \
--ph-phase-common-map <path_genetic_map> \
--ph-phase-common-config <path_config_txt_file> \
--ph-phase-common-sample-type <path_sample_type_txt_file> \
--output-directory <DIR> \
[options]
```

2. Ligate Common

```
dragen \
--enable-population-haplotype true \
--enable-ligate-common true \
--ph-ligate-common-input-list <path_to_txt_file> \
--output-directory <DIR> \
[options]
```

3. Phase Rare

```
dragen \
--enable-population-haplotype true \
--enable-phase-rare true \
--ph-phase-rare-input <path_to_preprocessed_file_output_of_step_1> \
--ph-phase-rare-input-region <string> \
--ph-phase-rare-scaffold <path_to_scaffold_file_output_of_step_1> \
--ph-phase-rare-scaffold-region <string> \
--ph-phase-rare-map <path_genetic_map> \
--ph-phase-rare-config <path_config_txt_file> \
--ph-phase-rare-sample-type <path_sample_type_txt_file> \
--output-directory <DIR> \
[options]
```

4. Concat All

To generate per chromosome haplotypes:

```
dragen \
--enable-population-haplotype true \
--enable-concat-all true \
--ph-concat-all-input-list <path_to_txt_file> \
--output-directory <DIR> \
[options]
```

To generate per genome haplotyped sites:

```
dragen \
--enable-population-haplotype true \
--enable-concat-all true \
--ph-concat-all-input-list-sites-only <path_to_txt_file> \
--output-directory <DIR> \
[options]
```

Input Files

msVCF Input (steps 1 and 3)

msVCF input list for the Phase Common step (step 1)

For the Phase Common step (step 1), it is recommended to provide msVCF generated with the DRAGEN gVCF Genotyper tool. This first step takes as input a .txt file with path to a single msVCF or a list of msVCF, one line per path. The msVCF must comply with the following requirements:

- per chromosome msVCF OR positionally sorted msVCF shards spanning a whole chromosome without overlap. See below for shard definition
- generated from the same reference build
- compressed and indexed
- with unphased GT calls
- with no duplicates
- all contigs of the studied genome must be listed in the header

Note: for mixed ploidy chromosomes each PAR and non-PAR regions of the chromosome must be treated as a single chromosome. For example, on human data, the sample input msVCF for chrX must be divided into chrX_par1, chrX_par2, and chrX_nonpar.

msVCF input for the Phase Rare step (step 3)

The msVCF input list provided at step 1 is pre-processed to generate a formatted msVCF called <prefix>.preprocess.vcf.gz. This formatted msVCF is generated in the directory and must be used as input of the Phase Rare step (step 3).

To facilitate parallel processing on distributed compute nodes, and to avoid overhead chromosome level multisample VCF download and upload per sub-chromosome processing, chromosome portions of equal size (shards) can be used as input. The gVCF Genotyper tool, with proper option, can generate these shards of equal size.

Note: streaming from the cloud is not supported. Instead use predownload and local input process to achieve maximum IO efficiency and stability.

Genetic map (steps 1 and 3)

A per chromosome genetic map corresponding to the studied species and to the reference build used for the msVCF input is required. You can use your own genetic map computed from the recombination rate of the species and its reference genome, or use the genetic map corresponding to the human hg38 reference genome available in the Imputation files accessible in the [Illumina DRAGEN Bio-IT Platform Support Site](#) page. DRAGEN does not generate custom genetic map files.

The genetic map should follow the format:

- 3 columns: position, chromosome number, distance (cM), in this order and tab separated
- Genetic map for mixed ploidy chromosome must be separated into as many PAR and non PAR regions (e.g. for human, chromosome X is split into PAR1 chrX_par1, PAR2 chrX_par2 and non PAR chrX_nonpar regions)
- Genetic map for region in which all samples are haploid is not needed (e.g. for human, chromosome Y chrY)

The user must ensure the genetic maps provided are from the same reference build than the reference used to generate the msVCF input.

Config file (steps 1 and 3)

This configuration file is a text file and is a required file. It allows for proper handling of haploid/diploid chromosomes and verification of concordance between genetic maps, msVCF input and sample type file information. Current configuration supports binary gender (male or female) and ploidy 2 or 1. When a region has different ploidies in male and female samples, the region is considered mixed ploidy region (e.g. for human, non PAR region on chromosome X chrX_nonpar).

You can use your own file, or use the file included in the `genetic_maps-hg38-2.0` folder available at [DRAGEN Bio-IT Platform support pages](#) on the Illumina website.

Example of a Config file

```
##version=1.0
##ref_build=hg38
filename      region    male_ploidy   female_ploidy
chr1.gmap.gz  chr1:1-248956422  2        2
chr2.gmap.gz  chr2:1-242193529  2        2
chr3.gmap.gz  chr3:1-198295559  2        2
chr4.gmap.gz  chr4:1-190214555  2        2
chr5.gmap.gz  chr5:1-181538259  2        2
chr6.gmap.gz  chr6:1-170805979  2        2
chr7.gmap.gz  chr7:1-159345973  2        2
chr8.gmap.gz  chr8:1-145138636  2        2
chr9.gmap.gz  chr9:1-138394717  2        2
chr10.gmap.gz chr10:1-133797422  2        2
chr11.gmap.gz chr11:1-135086622  2        2
chr12.gmap.gz chr12:1-133275309  2        2
chr13.gmap.gz chr13:1-114364328  2        2
chr14.gmap.gz chr14:1-107043718  2        2
chr15.gmap.gz chr15:1-101991189  2        2
chr16.gmap.gz chr16:1-90338345   2        2
chr17.gmap.gz chr17:1-83257441   2        2
chr18.gmap.gz chr18:1-80373285  2        2
chr19.gmap.gz chr19:1-58617616  2        2
chr20.gmap.gz chr20:1-64444167  2        2
chr21.gmap.gz chr21:1-46709983  2        2
chr22.gmap.gz chr22:1-50818468  2        2
chrX_par1.gmap.gz  chrX:1-2781479  2        2
chrX_nonpar.gmap.gz  chrX:2781480-155701382 1        2
chrX_par2.gmap.gz  chrX:155701383-156040895 2        2
```

Instructions to make a custom configuration file:

The config file is a text file with the headers:

- ##version
- ##ref_build indicating the reference build used for the study.

The Config file is a txt file and contains 4 columns, tabs delimited. Each of them must be populated.

Column information	Description
First column: filename	Specifies the genetic map basename, 1 name per line. Mixed ploidy chromosomes must be separated into par and non-par regions. Basenames must match genetic map basenames.
Second column: region	Specifies the start and end positions of the chromosome or sub-chromosome region with format <contig_name>:<start_position>-<end_position>. For chromosomes without mixed ploidy regions, the start position is 1, end position is the length of the chromosome (1-based, inclusive). For chromosomes with mixed ploidy regions, for each region, the start and end positions are those of the region (1-based, inclusive).
Third column: mixed ploidy subject	Specifies 2 on diploid chromosomes and PAR regions. 1 for non PAR region
Fourth column: diploid subject	Specifies 2 for all chromosomes

i | for mixed ploidy chromosome ensure the genetic map is separated into as many PAR and non-PAR regions with no overlap. Example: for human data prefix should be `chrX_par1`, `chrX_nonpar`, and `chrX_par2`.

Sample type file (steps 1 and 3)

The sample type file is a required file. The number of samples and name of samples in the input multisample VCF and sample type file should match.

The sample type file is a txt file with the following format

- 2 columns, tabs or space delimited
- First column: list of all sample names present in the input sample
- Second column: 1 or 2. 1 for subject with mixed ploidy chromosomes, 2 for subject with all diploid chromosomes.

Output Files

Phase Common step

The Phase common step (step 1) is run on a defined region, and outputs:

- A single scaffold msVCF and related msVCF index with phased common variants for that region. The default name is `dragen.ph_phase_common.vcf.gz`.
- A single formatted msVCF called `<prefix>.preprocess.vcf.gz` and related index. This formatted msVCF is generated in the directory and must be used as input of the Phase Rare step (step 3).

Ligate Common step

The Ligate Common step (step 2) ligates the regions phased in step 1 and outputs a single scaffold msVCF and related msVCF index with phased common variants for a single chromosome. The default name is `dragen.ph_ligate_common.vcf.gz`.

Phase Rare step

The Phase Rare step (step 3) is run on a defined region on a chromosome with preprocessed unphased msVCF from step 1 and phased scaffold msVCF from step 2, and outputs:

- A single phased msVCF and related msVCF index with phased common and rare variants for that region. The default name is `dragen.ph_rare_common.vcf.gz`.
- A single 8-column VCF and related index listing all sites that have been phased for that region. The default name is `dragen.ph_rare_common.sites.vcf.gz`. This output is used at the Concat-All step to generate a VCF file with all phased sites across the genome.

Concat All step

The Concat All processing is used to generate 2 types of output

1. Phased common and rare variants for a chromosome

The Concat All step (step 4) concatenates the regions phased in step 3 and outputs an msVCF and related index with phased common and rare variants for a single chromosome. The default name is `dragen.ph_concat_all.vcf.gz`.

2. List of phased sites

The output is useful for input in the ForceGT tool. The Concat All step lists all sites in a 8-column VCF format that have been phased and output a VCF and related index with list of phased sites. The output can be generated either from a list of phased site VCFs across the genome from step3, or in a second step once the list of per chromosome sites have been generated. The default name is `dragen.ph_concat_all.sites.vcf.gz`.

Command Line Options- Step 1: Phase Common

Option	Required	Description
--------	----------	-------------

--enable-population-haplotyping	Yes	Set to true to enable population haplotyping tool.
--enable-phase-common	Yes	Set to true to enable the Phase Common step.
--ph-phase-common-input-list	Yes	Provides a .txt file listing the sample input pertaining to one chromosome, with path to a single msVCF or a list of msVCF, one line per path. Note: in the case of mixed ploidy chromosome each PAR and non-PAR regions must be treated as a single chromosome.

--ph-phase-common-input-region	Yes	Specifies the target region to be phased. String in the format contigname: startposition-endposition. startposition-endposition is required for mixed ploidy chromosomes, but is optional for the other chromosomes. Regions must overlap between them for the downstream ligate common step. Examples of input region length for human data if not entire contig: 1 mbp or 10mbp (faster compute time with smaller regions) Note: in the case of chromosome with mixed ploidy regions and diploid regions, the command should be run with one region at a time (eg three runs with three regions, chrX_par1, chrX_nonpar and chrX_par2, instead of one run with region chrX).
--ph-phase-common-map	Yes	Provides path to the chromosome genetic map. Note: in the case of mixed ploidy chromosome, the genetic map name must be divided into PAR and non-PAR regions.
--ph-phase-common-config	Yes	Provides path to the txt config file.

--ph-phase-common-reference	No	Provides the path to a reference panel of haplotypes in msVCF format. Useful for iterative haplotyping to accelerate the process.
--ph-phase-common-scaffold	No	Provides the path to a scaffold of haplotypes in msVCF format. Useful for iterative haplotyping to accelerate the process.
--ph-phase-common-sample-type	Yes	Provides the path to the Sample type file.
--ph-phase-common-filter-maf	No	Default 0.001. Set the Minimum Allele Frequency threshold. All variants with allele frequency equal or above this MAF are phased during this Phase Common step.
--ph-phase-common-max-miss-gt-rate	No	Default 0.1. Set the threshold for variants to be skipped if the rate of missing GT is higher than this value.
-n --num-threads	No	Specifies the number of processor threads to use.
--output-directory	Yes	Specifies the output directory.
--output-file-prefix	No	Outputs filename with the defined prefix for the file generated by the pipeline.

Command Line Options- Step 2: Ligate Common

Option	Required	Description

--enable-population-haplotype	Yes	Set to true to enable population haplotyping tool.
--enable-ligate-common	Yes	Set to true to enable the Ligate Common step.
--ph-ligate-common-input-list	Yes	Provide a .txt file with list of phased msVCF related to a single chromosome. The msVCF provided are the output files of Phased Common step, in ascending order. Note: in the case of mixed ploidy chromosome each PAR and non-PAR regions must be treated as a single chromosome.
-n --num-threads	No	Specifies the number of processor threads to use.
--output-directory	Yes	Specifies the output directory.
--output-file-prefix	No	Outputs file name with the defined prefix for the file generated by the pipeline.

Command Line Options- Step 3: Phase Rare

Option	Required	Description
--enable-population-haplotype	Yes	Set to true to enable population haplotyping tool.
--enable-phase-rare	Yes	Set to true to enable the Phase Rare step.

--ph-phase-rare-input	Yes	Provides the path to the preprocessed unphased msVCF generated from Phase Common step covering the phase rare region.
--ph-phase-rare-input-region	Yes	Specifies the target region to be phased. String in the format contigname: startposition-endposition. startposition-endposition is required for mixed ploidy chromosomes, but is optional for the other chromosomes. Regions must overlap between them for the downstream ligate common step. Examples of input region length for human data if not entire contig: 1 mbp or 10mbp (faster compute time with smaller regions) Note: in the case of chromosome with mixed ploidy regions and diploid regions, the command should be run with one region at a time (eg three runs with three regions, chrX_par1, chrX_nonpar and chrX_par2, instead of one run with region chrX).

--ph-phase-rare-map	Yes	Provides the path to the genetic map of the chromosome. Note: in the case of mixed ploidy chromosome, the genetic map name must be divided into PAR and non-PAR regions.
--ph-phase-rare-config	Yes	Provides path to the txt config file.
--ph-phase-rare-scaffold	Yes	Provides the path to a scaffold of haplotypes in msVCF format generated from the Ligate Common step.
--ph-phase-rare-scaffold-region	Yes	Specify the scaffold region to be phased. String in the format contigname: startposition-endposition (startposition-endposition is optional). The scaffold region needs to cover the input region and allow buffer between regions.
--ph-phase-rare-sample-type	Yes	Provides the path to the Sample type file.
--ph-phase-rare-filter-maf	No	Default 0.001. Set the Maximum Allele Frequency threshold. All variants with allele frequency below this MAF are phased during this Phase Rare step. The value must be the same as the one provided by --ph-phase-common-filter-maf.
-n --num-threads	No	Specifies the number of processor threads to use.

--output-directory	Yes	Specifies the output directory.
--output-file-prefix	No	Outputs filename with the defined prefix for the file generated by the pipeline.

Command Line Options- Step 4: Concat All

Option	Required	Description
--enable-population-haplotyping	Yes	Set to true to enable population haplotyping tool.
--enable-concat-all	Yes	Set to true to enable the Concat All step.

--ph-concat-all-input-list	Yes when --ph-concat-all-input-list-sites-only is not provided	Provides a .txt file with list of phased msVCF pertaining to a single chromosome. The msVCF files provided are the output files of Phase Rare step, in ascending position order. Note: in the case of mixed ploidy chromosome each PAR and non-PAR regions must be treated as a single chromosome.
--ph-concat-all-input-list-sites-only	Yes when --ph-concat-all-input-list is not provided	Provides a .txt file with list of VCF containing all the haplotyped sites. The VCF files provided are the output files of Phase Rare step, in ascending position order, sex chromosomes at the end.

-n --num-threads	No	Specifies the number of processor threads to use.
--output-directory	Yes	Specifies the output directory.
--output-file-prefix	No	Outputs filename with the defined prefix for the file generated by the pipeline.

Population Haplotyping Accuracy

An additional module of the Population Haplotyping tool checks for the quality of the haplotypes produced based on a phased truth set provided as input.

Command Line Example

```
dragen \
--enable-population-haplotyping true \
--enable-phase-qc true \
--ph-phase-qc-validation <path_to_phased_truth_set> \
--ph-phase-qc-estimation <path_to_phased_msVCF> \
--ph-phase-qc-input-region <string> \
--output-directory <DIR> \
[options]
```

Command Line Options

Option	Required	Description
--enable-population-haplotyping	Yes	Set to true to enable population haplotyping tool.
--enable-phase-qc	Yes	Set to true to enable the quality control module.

--ph-phase-qc-validation	Yes	Provides the path to the phased truth set msVCF. Note: the validation msVCF must have the same samples as in the estimation msVCF for which the phasing accuracy is to be estimated.
--ph-phase-qc-estimation	Yes	Provides the path to the phased msVCF, output of Concat All to be validated.
--ph-phase-qc-input-region	Yes	Specifies the target region to be phased. String in the format contigname: startposition-endposition. startposition-endposition is optional.
-n --num-threads	No	Specifies the number of processor threads to use.
--output-directory	Yes	Specifies the output directory.
--output-file-prefix	No	Outputs filename with the defined prefix for the file generated by the pipeline.

Explify Pipeline

Explify Analysis Pipeline

The Explify Analysis Pipeline offers a dedicated informatics solution for the Illumina Respiratory Pathogen ID/AMR Enrichment Panel Kit (RPIP) and Illumina Urinary Pathogen ID/AMR Enrichment Panel Kit (UPIP).

The application delivers data analysis for simultaneous detection, quantification, and profiling of:

- More than 280 RNA and DNA respiratory pathogens, including SARS-CoV-2, Influenza, RSV, Mycobacterium and Legionella species, and >2000 antimicrobial resistance (AMR) markers for RPIP.

- More than 170 genitourinary pathogens, including fastidious, slow-growing, anaerobic uropathogens, sexually transmitted microorganisms, and more than 3,700 antimicrobial resistance (AMR) markers for UPIP.

Command Line Options

Option	Description
--enable-explify	Enables the Explify Pipeline. The default value is false
--output-file-prefix	Prefix for all output files for the pipeline
--output-directory	Directory for all output files
--explify-sample-list	Input sample list .tsv file with sample IDs, FASTQs, etc.
--explify-test-panel-name	Typically set to "RPIP" or "UPIP" for panel name
--explify-test-panel-version	Set to test panel version. For example: 7.3.2
--explify-ref-db-dir	Path to root directory for Explify Database files
--intermediate-results-dir	Optional Area for temporary files. Size must be greater than size of all FASTQ files multiplied by 3
--explify-load-db-ram	Optional Load the database into RAM if it is not on ram disk. The default value is false.
--explify-no-read-qc	Optional Turn off read QC on FASTQs before analysis. The default value is false
--explify-internal-control	Optional Set the internal control from an accepted list. The default value is Enterobacteri phage T7.
--explify-internal-control-concentration	Optional Set the internal control concentration. The default value is 12100000.
--explify-ncpus	Optional Set the number of CPUs available for processing

Example Command Line

```
/opt/edico/bin/dragen \
--enable-explify=true \
--output-file-prefix <PREFIX> \
--explify-sample-list /path/to/sample/list/tsv \
--explify-test-panel-name <"RPIP"/"UPIP"> \
--explify-test-panel-version <VERSION> \
--explify-ref-db-dir /path/to/root/db/dir \
--explify-load-db-ram=true \
--output-directory <OUTPUT_DIR> \
--intermediate-results-dir <OUTPUT_DIR> \
--explify-ncpus=1
```

Input Files

Sample Input List

Applies to:

- `--explify-sample-list`

The sample input list is a column-formatted file with tab separations between the columns (ie, a TSV file).

SampleID	BatchID	RunID	ControlFlag	FastQs
MySample	MyBatch	MyRun	POS	/path/to/fastq1.g
				z

- The `SampleID` must be unique
- `BatchID` and `RunID` help track and manage sample analysis
 - The `BatchID` can be used to track libraries that were prepared together, and the `RunID` is used to track sequencing runs. They can also be left blank.
- The `ControlFlag` value can be `POS`, `NEG`, `BLANK`, or left empty
 - `POS` is used to indicate a positive control sample
 - `NEG` is used to indicate a negative control sample
 - `BLANK` is used to indicate a blank control sample (eg buffer only)
- Multiple FASTQ files are tab delimited

Internal Control

Applies to:

- `--explify-internal-control`
- `--explify-internal-control-concentration`

An internal control can be specified using the following input controls. The values are case sensitive. If `NONE` is specified, the internal control concentration is ignored. The default is `Enterobacteria phage T7`.

- `Allobacillus halotolerans`
- `Armored RNA Quant Internal Process Control`
- `Enterobacteria phage T7 Default`
- `Escherichia virus MS2`
- `Escherichia virus Qbeta`
- `Escherichia virus T4`
- `Imtechella halotolerans`
- `Phocid alphaherpesvirus 1`
- `Phocine morbillivirus`
- `Truepera radiovictrix`
- `NONE`

The internal control concentration is an integer representing the number of copies/mL of sample for the internal control.

Reference Databases

Applies to:

- `--explify-ref-db-dir`
- `--explify-test-panel-name`
- `--explify-test-panel-version`
- `--explify-load-db-ram`

The Explify Reference Databases are required to run the Explify Analysis Pipeline in DRAGEN and are not part of the core DRAGEN distribution.

Preparation

Before downloading the databases you will need to create a directory with at least 150 GB available. The databases will be stored in the directory and referenced with the `-d` parameter.

Download Script

Download and management of Explify databases is handled by a shell script. The script can be downloaded with the following command:

```
wget -O explify-dbs.sh https://illumina-explify-databases.s3.us-east-1.amazonaws.com/explify-dbs.sh
chmod +x explify-dbs.sh
```

Search for Available Databases to Download

The `search` subcommand lists the databases available for download:

Option	Description
<code>-d</code>	Database directory used for storage
<code>-p</code>	Panel name
<code>-n</code>	Number of CPUs that can be used to download the files (defaults to 1)

```
$ ./explify-dbs.sh search -d explify-databases/
2 database(s) found meeting those criteria:
- RPIP-5.11.7
- UPIP-7.3.7
```

Download a Database

The `download` subcommand with the following options, downloads database files for a panel:

Option	Description
<code>-d</code>	Database directory used for storage
<code>-p</code>	Panel name
<code>-v</code>	Panel version
<code>-n</code>	Number of CPUs that can be used to download the files (defaults to 1)

The following example will:

- Download the UPIP-7.3.7 files
- Download the required files to the `common` subdirectory
- Run the checksums

It is best to run the command by `screen` or `nohup` due to the size of the files.

```
./explify-dbs.sh download -d explify-databases/ -p UPIP -v 7.3.7 -n 20
```

List Downloaded Databases

The `list` subcommand is used to view the databases that have already been downloaded with the following options:

Option	Description
<code>-d</code>	Database directory used for storage
<code>-p</code>	Panel name

```
$ ./explify-dbs.sh list -d explify-databases/
```

Database Integrity

The `download` subcommand automatically checks the file checksums after the databases are downloaded. The `check` subcommand can be used as a standalone command to check the files with the following options:

Option	Description
<code>-d</code>	Database directory used for storage
<code>-p</code>	Panel name
<code>-v</code>	Panel version
<code>-n</code>	Number of CPUs that can be used to download the files (defaults to 1)

```
$ ./explify-dbs.sh check -d explify-databases/ -p UPIP -v 7.3.7 -n 20
```

Using the Databases

- If the database file is stored on a normal file system, set `--explify-load-db-ram=true` to load the database file into memory for faster analysis
- If the database file is stored on a RAM disk, set `--explify-load-db-ram=false` to reduce the load time over many Explify Pipeline runs

The default location of the databases is `/Explify-databases`, and has the following folder structure when extracted:

```
explify-databases/
  RPIP/
```

```
5.11.7/
UPIP/
7.3.7/
```

The following example shows the command lines to run an analysis with RPIP 5.11.7:

```
--explify-ref-db-dir /explify-databases
--explify-test-panel-name RPIP
--explify-test-panel-version 5.11.7
```

Output

The output of the Explify Analysis Pipeline is a `report.json` file written to the specified output directory. It is similar to the JSON reports available in BaseSpace applications, but contains more information. In contrast with the Explify BaseSpace App, all the information is in the JSON file. There is not a separate PDF, consensus FASTA files, or convenient XLSX file.

Some information available in BaseSpace reports is not available in the core DRAGEN Explify pipeline, including Pangolin SARS-CoV-2 lineage assignment (RPIP), viral AMR marker reporting (RPIP), organism relative abundance, footnotes/abbreviations/interpretive data, and clinical tertiary drug and drug class resistance results (intrinsic or acquired).

Report.json format

Top-Level Node

The fields in the top-level node of the JSON report provide general metadata and version information.

Field	Description
<code>.accession</code>	Sample identifier
<code>.deploymentEnvironment</code>	Environment in which the results were produced
<code>.batchId</code>	Identifier used for a batch of samples prepared in the lab at the same time
<code>.analysisId</code>	Identifier for the Explify analysis
<code>.runId</code>	Identifier used for a sequencing run

Field	Description
.controlFlag	Indicates whether the sample is a control. It is based on the ControlFlag field in the sample TSV and can be set to POS, NEG, BLANK, or -
.dragenVersion	DRAGEN release version
.analysisPipelineVersion	Analysis pipeline version
.testType	RPIP or UPIP
.testVersion	Version of the test
.testName	Name of the test, eg Explify® Respiratory Pathogen ID/AMR Panel (RPIP) – Data Analysis Solution
.testUse	For Research Use Only. Not for use in diagnostic procedures.
.reportTime	Time the report was generated

.qcReport Node

The fields are relative to .qcReport. This section provides information about the FASTQ file before and after read trimming.

Field	Description
.sampleQc	Read QC information
.sampleQc.entropy	Kmer entropy of reads after read QC processing
.sampleQc.gContent	Proportion of guanine (G) base calls in reads after read QC processing
.sampleQc.libraryQScore	Quality score of the library after read QC processing
.sampleQc.postQualityMeanReadLength	Average read length after read QC processing

Field	Description
.sampleQc.postQualityReads	Number of reads in sample after QC processing
.sampleQc.preQualityMeanReadLength	Average read length before read QC processing
.sampleQc.totalRawReads	Number of reads in sample before read QC processing
.sampleQc.uniqueReads	Number of unique reads in sample before read QC processing
.sampleQc.uniqueReadsProportion	Proportion of unique reads in sample before read QC processing

.qcReport.sampleComposition Node

All of the fields are relative to .qcReport.sampleComposition. This section provides information about the composition of the sample.

Field	Description
.readClassification	Proportion of reads classified to the following groups:
.readClassification.targeted	Targeted reference sequences
.readClassification.untargeted	Untargeted reference sequences
.readClassification.ambiguous	More than one pathogen class
.readClassification.unclassified	Could not be classified
Field	Description
.targeted	Proportion of targeted reads classified to the following groups:
.targeted.viral	Viral targeted sequences
.targeted.bacterial	Bacterial targeted sequences
.targeted.fungal	Fungal targeted sequences
.targeted.parasitic	Parasitic targeted sequences
.targeted.amr	Bacterial AMR targeted sequences

Field	Description
.targeted.internalControl	Internal Control (IC) targeted sequences
Field	Description
.untargeted	Proportion of untargeted reads classified to the following groups:
.untargeted.viral	Viral untargeted sequences
.untargeted.bacterial	Bacterial untargeted sequences
.untargeted.fungal	Fungal untargeted sequences
.untargeted.parasitic	Parasitic untargeted sequences
.untargeted.amr	Bacterial AMR untargeted sequences
.untargeted.internalControl	Internal Control (IC) untargeted sequences
.untargeted.human	Human sequences
.untargeted.lowComplexity	Sequences with low complexity in base composition (eg poly-A tails)

.qcReport.internalControls Node

The internalControls object is a list that gives the name and RPKM for the 10 commercially available spike-in control options. Refer to the following code block:

```
[
{
  "name": "Allobacillus halotolerans",
  "rpkm": 0
},
{
  "name": "Armored RNA Quant Internal Process Control",
  "rpkm": 0
},
{
  "name": "Enterobacteria phage T7",
  "rpkm": 180323
},
```

```
{  
  "name": "Escherichia virus MS2",  
  "rpkm": 0  
},  
{  
  "name": "Escherichia virus Qbeta",  
  "rpkm": 0  
},  
{  
  "name": "Escherichia virus T4",  
  "rpkm": 0  
},  
{  
  "name": "Imtechella halotolerans",  
  "rpkm": 0  
},  
{  
  "name": "Phocid alphaherpesvirus 1",  
  "rpkm": 0  
},  
{  
  "name": "Phocine morbillivirus",  
  "rpkm": 0  
},  
{  
  "name": "Truepera radiovictrix",  
  "rpkm": 0  
}  
]
```

Example

```
[  
  {  
    "name": "Allobacillus halotolerans",  
    "rpkm": 0  
}
```

```
 },
{
  "name": "Armored RNA Quant Internal Process Control",
  "rpkm": 0
},
{
  "name": "Enterobacteria phage T7",
  "rpkm": 180323
},
{
  "name": "Escherichia virus MS2",
  "rpkm": 0
},
{
  "name": "Escherichia virus Qbeta",
  "rpkm": 0
},
{
  "name": "Escherichia virus T4",
  "rpkm": 0
},
{
  "name": "Imtechella halotolerans",
  "rpkm": 0
}
```

```

    },
    {
        "name": "Phocid alphaherpesvirus 1",
        "rpkm": 0
    },
    {
        "name": "Phocine morbillivirus",
        "rpkm": 0
    },
    {
        "name": "Truepera radiovictrix",
        "rpkm": 0
    }
]

```

.userOptions Node

The fields are relative to .userOptions.

Field	Description
.quantitativeInternalControlName	Indicates the Internal Control used for absolute quantification (recommendation: Enterobacteria phage T7)
.quantitativeInternalControlConcentration	Internal Control concentration used in absolute quantification calculations (recommendation: 1.21×10^7 copies/mL)

Field	Description
.readQcEnabled	Boolean field that indicates whether read trimming was enabled

.targetReport.microorganisms Node

The fields are relative to .targetReport.microorganisms. The value of the microorganisms field is an array of objects describing microorganism detection metrics and metadata. The following table describes a microorganism's array of objects.

Field	Description
.class	Microorganism class (viral, bacterial, fungal, parasite)
.name	Name of detected microorganism
.coverage	Proportion of targeted microorganism sequence bases that appear in sequencing reads
.ani	Average nucleotide identity of majority consensus sequence to targeted microorganism reference sequences
.medianDepth	Median depth of reads aligned to targeted microorganism reference sequences, indicating the median number of times each targeted microorganism sequence base appears in sequencing reads
.condensedDepthVector	Read depth across the targeted microorganism reference sequences, concatenated and condensed (if needed) down to 256 items.

Field	Description
.rpkm	Normalized representation of the number of reads aligned to targeted microorganism reference sequences (aligned reads per kilobase of targeted sequence per million reads)
.alignedReadCount	Number of reads aligned to reference genome (or segment)
.kmerReadCount	Number of reads assigned to targeted microorganism reference sequences by k-mer classification
.absoluteQuantityRatio	Numerical absolute quantification value
.absoluteQuantityRatioFormatted	Formatted absolute quantification value and units
.phenotypicGroup	Grouping indicating general association with normal flora, colonization, or contamination from the environment or other sources, as well as general association with disease
Field	Description
.associatedAmrMarkers	Information about the detected and predicted AMR markers associated with this bacterium. Only present for bacteria.
.associatedAmrMarkers.detected	A list of the detected AMR markers associated with this bacterium. Only present for bacteria.
.associatedAmrMarkers.predicted	A list of the predicted AMR markers associated with this bacterium. Only present for bacteria.

Field	Description
.consensusGenomeSequences	(RPIP viruses only) Information about genome (or segment) consensus sequence
.consensusGenomeSequences.sequence	The consensus genome (or segment) sequence
.consensusGenomeSequences.referenceAccession	Accession of the reference genome (or segment)
.consensusGenomeSequences.referenceDescription	Description of the reference genome (or segment)
.consensusGenomeSequences.referenceLength	The length of the reference genome
.consensusGenomeSequences.maximumAlignmentLength	Longest contiguous alignment between consensus and reference genome (or segment)
.consensusGenomeSequences.maximumGapLength	Longest contiguous gap between consensus and reference genome (or segment)
.consensusGenomeSequences.coverage	Proportion of reference genome sequence bases that appear in sequencing reads
.consensusGenomeSequences.ani	Average nucleotide identity (ANI) of majority consensus sequence to reference genome (or segment)
.consensusGenomeSequences.alignedReadCount	Number of reads aligned to targeted microorganism reference genome sequences
.consensusGenomeSequences.medianDepth	Median depth of reads aligned to reference genome (or segment), indicating the median number of times each reference genome (or segment) base appears in sequencing reads

Field	Description
.consensusGenomeSequences.targetAnnotation	List of target annotations for the reference genome (or segment). Each annotation is a JSON object with the following fields: start (int), end (int), strand (string), target_name (string), type (string).
Field	Description
.consensusTargetSequences	(RPIP viruses only) Information about targeted region consensus sequence
.consensusTargetSequences.sequence	Targeted region consensus sequence
.consensusTargetSequences.name	Targeted region name
.consensusTargetSequences.referenceAccession	Targeted region reference accession
.consensusTargetSequences.depthVector	Read depth across the targeted region
Field	Description
.explifyInterpretation	Information about Explify's automated interpretation results
.explifyInterpretation.predictedPresent	Explify prediction that microorganism is present (true/false)
.explifyInterpretation.notes	List of notes about the Explify prediction result
.explifyInterpretation.subpanels	List of subpanels that microorganism belongs to
.explifyInterpretation.relatedOrganisms	Object that gives key metrics for closely related on- and off-panel microorganisms that were detected. See below for details.

.targetReport.microorganisms.relatedOrganisms Node

The relatedOrganisms object includes a list of the organisms that were considered as part of this organism's interpretation. The fields below describe an object in the relatedOrganisms array.

Field	Description
.name	Related microorganism's name
.onPanel	Whether the related microorganism is on the panel or not
.kmerReadCount	The number of reads assigned to the microorganism using a k-mer based approach. This field is only present when this approach is applied. Currently, it is present for UPIP but not RPIP
.coverage	The coverage to the microorganism resulting from alignment
.ani	The ANI to the microorganism resulting from alignment
.alignedReadCount	Number of reads aligned to related microorganism reference sequences

.targetReport.microorganisms.variants Node

The fields are relative to .targetReport.microorganisms.variants. The variants object is only present for select viruses.

Field	Description
.referenceAccession	NCBI accession of reference sequence used for variant calling
.segment	(Influenza A only). Segment number of reference sequence
.ntChange	Nucleotide change associated with the variant

Field	Description
.referencePosition	Variant position in reference sequence
.referenceAllele	Reference allele at same position as the variant
.variantAllele	Variant allele
.depth	Variant depth, indicating the number of times the variant appears in sequencing reads.
.alleleFrequency	Frequency of the variant allele in the sequencing reads

.targetReport.amrMarkers Node

The fields are relative to .targetReport.amrMarkers. This section provides information about the detected bacterial AMR markers.

Field	Description
.class	Microorganism class (eg bacterial)
.modelType	AMR marker detection model specified by CARD (homolog, protein variant, rRNA variant)
.geneFamily	AMR marker family name in CARD
.name	AMR marker name
.referenceAccession	NCBI or CARD accession of AMR marker reference sequence
.coverage	Proportion of reference genome (or segment) bases that appear in sequencing reads (protein alignment for homolog and protein variant model types; DNA alignment for rRNA variant model type)

Field	Description
.pid	Percent identity of majority consensus sequence aligned to reference sequence (protein alignment for homolog and protein variant model types; DNA alignment for rRNA variant model type)
.medianDepth	Median depth of reads aligned to AMR marker reference sequence, indicating the median number of times each AMR marker sequence residue appears in sequencing reads (protein alignment for homolog and protein variant model types; DNA alignment for rRNA variant model type)
.rpkm	Median depth of reads aligned to AMR marker reference sequence, indicating the median number of times each AMR marker sequence residue appears in sequencing reads (protein alignment for homolog and protein variant model types; DNA alignment for rRNA variant model type)
.alignedReadCount	The read count to the marker resulting from alignment
.nucleotideConsensusSequence	(UPIP only) The nucleotide consensus sequence
.proteinConsensusSequence	(UPIP only) The protein consensus sequence
.nucleotideDepthVector	The depths across the nucleotide alignment, not condensed
.proteinDepthVector	The depths across the protein alignment, not condensed

Field	Description
.associatedMicroorganisms	Lists of the detected and predicted organisms associated with this marker
.associatedMicroorganisms.all	A list of all organisms associated with this marker
.associatedMicroorganisms.detected	A list of the detected organisms associated with this marker
.associatedMicroorganisms.predicted	A list of the predicted organisms associated with this marker
.explifyInterpretation	Information about Explify's automated interpretation results
.explifyInterpretation.predictedPresent	Whether Explify interpretation predicts that the marker is present (true/false)
.explifyInterpretation.confidence	Whether the AMR marker is predicted with high or medium confidence
.explifyInterpretation.notes	List of notes about the interpretation result

.targetReport.amrMarkers.variants Node

The fields are relative to targetReport.amrMarkers.variants. This section provides information about variants detected on select bacterial AMR markers.

Field	Description
.category	"Bacterial Variant; Known AMR"
.referenceSourceMicroorganism	Microorganism that reference sequence is associated with in NCBI
.product	The protein product of the gene
.ntChange	The nucleotide change
.referencePosition	The position on the reference sequence

Field	Description
.referenceAllele	The reference sequence at the position of the variant
.variantAllele	The variant sequence
.depth	The depth at the variant position
.alleleFrequency	The frequency of the variant allele in the read pile up
.annotation	Type of change (eg "Nonsynonymous Variant")
.aaChange	Amino acid change
.epistaticGroups	List of epistatic groups the variant is associated with

Kmer Classifier

The metagenomics classifier uses a k-mer based classification algorithm to classify each query sequence, usually a read, against a collection of reference sequences. The reference sequences need to be indexed into a database searchable by query sequences and classified to taxid(s) associated with the reference sequences.

Command Line Options

Option	Description
--enable-kmer-classifier	Enables the Kmer Classifier. The default value is false
--output-file-prefix	Prefix for all output files for the classifier
--output-directory	Directory for all output files
--kmer-classifier-input-read-file	Input sequence file (zipped or unzipped) to the Kmer Classifier
--kmer-classifier-db-file	Database of sequences to classify against
--intermediate-reuslts-dir	Optional Area for temporary files. Size must be greater than size of all FASTQ files multiplied by 2

Option	Description
--kmer-classifier-load-db-ram	Optional Load the database onto RAM. Do not use if database is on ramdisk. The default value is false
--kmer-classifier-multiple-inputs	Optional Set to true to run with multiple inputs where input .tsv file has two columns: ID, Inputfile. The default value is false.
--kmer-classifier-min-window	Optional The minimum number of consecutive kmers for classify assignment at taxid. The default value is 1
--kmer-classifier-output-read-seq	Optional Enable a read sequence column in the output file. The default value is false.
--kmer-classifier-output-taxid-seq	Optional Enable a taxid string column in the output file. The default value is false.
--kmer-classifier-db-to-taxid-json	Optional Path to JSON file that maps database IDs to external taxids, names, and ranks
--kmer-classifier-no-read-output	Optional Do not create an individual read output. The default value is false
--kmer-classifier-no-taxid-counts	Optional Do not write a taxid count output file. The default value is false
--kmer-classifier-protein-input	Optional Indicate protein query sequences and database. The default value is false
--kmer-classifier-reads-to-keep	Optional Path to file of read IDs to use in alignment with one read per line
--kmer-classifier-no-remove-dups	Optional Do not deduplicate reads in input files
--kmer-classifier-ncpus	Optional Set the number of CPUs available for processing

Example Command Line

```
/opt/edico/bin/dragen \
--enable-kmer-classifier=true \
--output-file-prefix <PREFIX> \
--output-directory <OUTPUT_DIR> \
--kmer-classifier-input-read-file /path/to/fastq.gz \
--kmer-classifier-db-file /path/to/database \
--kmer-classifier-min-window 1 \
--kmer-classifier-ncpus=2 \
--kmer-classifier-output-read-seq=false \
--kmer-classifier-output-taxid-seq=false
```

Example

```
/opt/edico/bin/dragen \
--enable-kmer-classifier=true \
--output-file-prefix <PREFIX> \
--output-directory <OUTPUT_DIR> \
--kmer-classifier-input-read-file /path/to/fastq.gz \
--kmer-classifier-db-file /path/to/database \
--kmer-classifier-min-window 1 \
--kmer-classifier-ncpus=2 \
--kmer-classifier-output-read-seq=false \
--kmer-classifier-output-taxid-seq=false
```

Input Files

Input Reads

Applies to:

- --kmer-classifier-input-read-file
- --kmer-classifier-multiple-inputs

If the analysis is for a single FASTA/FASTQ read file, then the file name is input to `--kmer-classifier-input-read-file` and `--kmer-classifier-multiple-inputs=false`. Multiple read files can be submitted to the Kmer Classifier at one time to minimizing the load time for a large reference sequence database. To submit multiple read files the input file must be a `.tsv` file with two columns. The first column is a unique ID and the second column is the path to the read file. The ID is used to distinguish the output files. There is no header line. The `.tsv` file is the file input to `--kmer-classifier-input-read-file` and `--kmer-classifier-multiple-inputs=true`.

Reference Sequences

Applies to:

- `--kmer-classifier-db-file`
- `--kmer-classifier-db-to-taxid-json`
- `--kmer-classifier-load-db-ram`

A file of reference sequences (the "database") can be quite large. However, given their size they will not be part of the core DRAGEN distributable. There will be a separate mechanism for the client to obtain reference databases.

- If the database file is stored on a normal file system, set `--kmer-classifier-load-db-ram=true`. This will tell the Kmer Classifier to load the database file into memory for faster analysis.
- If the database file is stored on a RAM disk, set `--kmer-classifier-load-db-ram=false`. This will reduce the load time over many Kmer Classifier runs.

DB TaxID JSON Mapping File

Applies to:

- `--kmer-classifier-db-to-taxid-json`

When the reference sequence database is downloaded, it will include the input file. The file associates a taxid internal to the classifier database, to an external source (similar to the NCBI taxonomy). The JSON file is a dictionary where the keys are internal taxids, and mapped to external taxid, name, and rank.

See the following example:

```
{
"2": {"taxid": 2, "name": "bacteria", "rank": "kingdom"},
"3": {"taxid": 2697049, "name": "SARS-CoV-2", "rank": "subspecies"},
"4": {"taxid": 5052, "name": "Aspergillus", "rank": "genus"}
}
```

Reference Databases

Applies to:

- `--kmer-classifier-db-file`

- --kmer-classifier-load-db-ram

The Reference Sequence Databases are required to run the Kmer Analysis in DRAGEN and are not part of the core DRAGEN distribution.

- If the database file is stored on a normal file system, set --kmer-classifier-load-db-ram=true to load the database file into memory for faster analysis
- If the database file is stored on a RAM disk, set --kmer-classifier-load-db-ram=false to reduce the load time over many Kmer Classifier runs.

Genome Database

The Genome Database includes internally curated whole genomes for human, bacteria, viruses, and fungi. The NCBI taxonomy from August 25, 2020 was used to build the database, and the sequences were collected in August 2022.

The following command downloads the reference index file and the taxid mapping JSON:

```
wget https://illumina-explify-databases.s3.us-east-1.amazonaws.com/kmer-
classifier/dragen-kmer-classifier.genomes.v5dh.t5db
wget https://illumina-explify-databases.s3.us-east-1.amazonaws.com/kmer-
classifier/dragen-kmer-classifier.genomes.name_map.json
```

NT Database

The database Includes the human, bacterial, viral, and fungal subsets of the NCBI nucleotide database. The sequences were clustered at 0.95 PID before building the reference index. The NCBI taxonomy from April 24, 2023 was used to build the database.

The following command downloads the reference index file and the taxid mapping JSON:

```
wget https://illumina-explify-databases.s3.us-east-1.amazonaws.com/kmer-
classifier/dragen-kmer-classifier.nt_95.v5dh.t5db
wget https://illumina-explify-databases.s3.us-east-1.amazonaws.com/kmer-
classifier/dragen-kmer-classifier.nt_95.name_map.json
```

UniRef90 Database

The database Includes human, bacterial, viral, and fungal subsets of the UniRef90 database. The NCBI taxonomy from April 24, 2023 was used to build the database.

The following command downloads the reference index file and the taxid mapping JSON:

```
wget https://illumina-explify-databases.s3.us-east-1.amazonaws.com/kmer-
classifier/dragen-kmer-classifier.u90.v5dh.t5db
```

```
wget https://illumina-explify-databases.s3.us-east-1.amazonaws.com/kmer-
classifier/dragen-kmer-classifier.u90.name_map.json
```

Reads to Keep File

Applies to:

- `--kmer-classifier-reads-to-keep`

This optional input file tells the classifier to only work with a subset of the input reads. The reads are indexed 0 to n-1 (where n is the number of reads in the input read file, in order, from beginning to end). This file is a simple input file with one index per line.

The following example only keeps the first 4 reads of a file:

```
0
1
2
3
```

Output

There are two output files, one organized around the reads, and the other organized around the taxids.

Read-level Output

Applies to:

- `--kmer-classifier-output-taxid-seq`
- `-kmer-classifier-output-read-seq`

The main output is a TSV file with the extension `.classifier.tsv` and the following layout:

- Does not have a header line
- Does have tab-separated columns
- Number of columns vary depending on command line options

The columns are as follows:

Column	Data Type	Description
1	[integer]	Read index
2	[integer]	Taxid the read classified to
3	[integer]	Maximum number of contiguous kmers that classified to this taxid

Column	Data Type	Description
4	[integer]	Score assigned to the classification
5	[integer]	Number of kmers that classified to this taxid
6	[integer]	Read duplication count
7	[list of integers separated by comma]	Taxid that each kmer classified to Requires --kmer-classifier-output-taxid-seq flag to be set.
8	[string]	Read Sequence Requires --kmer-classifier-output-read-seq flag to be set

TaxID-level Output

The second output file is a TSV file with the extension `.classifier.taxid_kmer_counts.tsv` and the following layout:

- Does have a header line
- Does have tab-separated columns

The columns are as follows:

Header	Data Type	Description
db_taxid	[integer]	Identifier for this taxid used internally in the database
duplicity	[float]	Ratio of total number of kmers from reads assigned to the taxid compared to the number of distinct kmers from reads assigned to the taxid
distinct_coverage	[integer]	Percent of kmers in the database assigned to the taxid that are covered by kmers in the reads assigned to the taxid
read_count	[integer]	Number of reads that are classified to the taxid
total_kmer_count	[integer]	Number of kmers that are classified to the taxid

Header	Data Type	Description
distinct_kmer_count	[integer]	Number of distinct kmers that are classified to the taxid
cumulative_read_count	[integer]	Cumulative number of reads assigned to the taxid and its taxonomic descendants
taxid	[integer]	Taxid
name	[string]	Name associated with the taxid
rank	[string]	Taxonomic rank of the taxid
taxid_distinct_kmer_count	[string]	Number of distinct kmers assigned to the taxid from the reference sequences

Recipes

This section contain recommended command line options for specific DRAGEN pipelines. For an overview of DRAGEN command line parsing, see [Multicaller Workflows on page 391](#) for more information.

Germline WES

This recipe is for processing whole exome sequencing data for germline workflows.

Example Command Line

For most scenarios, simply creating the union of the command line options from the single caller scenarios will work.

- Configure the INPUT options
- Configure the OUTPUT options
- Configure MAP/ALIGN depending on if realignment is desired or not
- Configure the VARIANT CALLERs based on the application
- Configure any additional options
- Build up the necessary options for each component separately, so that they can be re-used in the final command line

The following are partial templates that can be used as starting points. Adjust them accordingly for your specific use case.

```
#!/bin/bash

set -euo pipefail


# Path to DRAGEN hashtable
DRAGEN_HASH_TABLE=<REF_DIR>

# Path to output directory for the DRAGEN run
OUTPUT=<OUT_DIR>

# File prefix for DRAGEN output files
PREFIX=<OUT_PREFIX>

# Define the input sources, select fastq list, fastq, bam, or cram.
INPUT_FASTQ_LIST="

--fastq-list $FASTQ_LIST \
--fastq-list-sample-id $FASTQ_LIST_SAMPLE_ID \
"

INPUT_FASTQ="

--fastq-file1 $FASTQ1 \
--fastq-file2 $FASTQ2 \
--RGSM $RGSM \
--RGID $RGID \
"

"
```

```
INPUT_BAM="

--bam-input $BAM \
"

INPUT_CRAM="

--cram-input $CRAM \
"

# Select input source, here in this example we use INPUT_FASTQ_LIST

INPUT_OPTIONS="

--ref-dir $DRAGEN_HASH_TABLE \
$INPUT_FASTQ_LIST \
"

OUTPUT_OPTIONS="

--output-directory $OUTPUT \
--output-file-prefix $PREFIX \
"

MA_OPTIONS="

--enable-map-align true \
--enable-sort true \
--enable-duplicate-marking true \
"
```

```
"  
  
CNV_OPTIONS=""  
  
--enable-cnv true \  
  
--cnv-target-bed $CNV_TARGET_BED \  
  
--cnv-normals-list $CNV_PANEL_OF_NORMALS \  
  
"  
  
  
SNV_OPTIONS=""  
  
--enable-variant-caller true \  
  
--vc-target-bed $VC_TARGET_BED \  
  
"  
  
  
SV_OPTIONS=""  
  
--enable-sv true \  
  
--sv-exome true \  
  
--sv-call-regions-bed $SV_TARGET_BED \  
  
"  
  
  
# Construct final command line  
  
CMD=""  
  
dragen \  
  
$INPUT_OPTIONS \  
  
$OUTPUT_OPTIONS \  

```

```
$MA_OPTIONS \
$CNV_OPTIONS \
$SNV_OPTIONS \
$SV_OPTIONS \
"
# Execute
echo $CMD
bash -c $CMD
```

Optional Settings

Optional settings per component are listed below. Refer to [Command-line Options on page 54](#) for a full list of options.

CNV

Include the matched normal sample in the CNV panel of normals

Option	Description
--cnv-enable-gcbias-correction true	Enable or disable GC bias correction when generating target counts. For more information, refer to GC Bias Correction on page 186 .

Generating Panel of Normals (PON)

Somatic WES CNV requires PON files. The following is an example command of PON generation. Options used for panel of normals generation (BED file, GC Bias Correction, etc) should be matched when processing the case sample. Refer to [Panel of Normals on page 188](#) for more information.

```
CNV_PON_OPTIONS="
```

```
--enable-cnv true \
--cnv-target-bed $CNV_TARGET_BED \
"
CMD="

dragen \
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \
$CNV_PON_OPTIONS \
"
"
```

Germline WGS

This recipe is for processing whole-genome sequencing data for germline workflows.

Example Command Line

For most scenarios, simply creating the union of the command line options from the single caller scenarios will work.

- Configure the INPUT options
- Configure the OUTPUT options
- Configure MAP/ALIGN depending on if realignment is desired or not
- Configure the VARIANT CALLERs based on the application
- Configure any additional options
- Build up the necessary options for each component separately, so that they can be re-used in the final command line

The following are partial templates that can be used as starting points. Adjust them accordingly for your specific use case.

```
#!/bin/bash
```

```
set -euo pipefail

# Path to DRAGEN hashtable
DRAGEN_HASH_TABLE=<REF_DIR>

# Path to output directory for the DRAGEN run
OUTPUT=<OUT_DIR>

# File prefix for DRAGEN output files
PREFIX=<OUT_PREFIX>

# Define the input sources, select fastq list, fastq, bam, or cram.
INPUT_FASTQ_LIST="

--fastq-list $FASTQ_LIST \
--fastq-list-sample-id $FASTQ_LIST_SAMPLE_ID \
"

INPUT_FASTQ="

--fastq-file1 $FASTQ1 \
--fastq-file2 $FASTQ2 \
--RGSM $RGSM \
--RGID $RGID \
"

"
```

```
INPUT_BAM="

--bam-input $BAM \
"

INPUT_CRAM="

--cram-input $CRAM \
"

# Select input source, here in this example we use INPUT_FASTQ_LIST

INPUT_OPTIONS="

--ref-dir $DRAGEN_HASH_TABLE \
$INPUT_FASTQ_LIST \
"

OUTPUT_OPTIONS="

--output-directory $OUTPUT \
--output-file-prefix $PREFIX \
"

MA_OPTIONS="

--enable-map-align true \
--enable-sort true \
--enable-duplicate-marking true \
"
```

```
CNV_OPTIONS="

--enable-cnv true \
--cnv-enable-self-normalization true \
"

SNV_OPTIONS="

--enable-variant-caller true \
"

SV_OPTIONS="

--enable-sv true \
"

PARALOG_OPTIONS="

--enable-cyp2b6 true \
--enable-cyp2d6 true \
--enable-gba true \
--enable-smn true \
--repeat-genotype-enable true \
"

# Construct final command line

CMD="
```

```
dragen \
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \
$MA_OPTIONS \
$CNV_OPTIONS \
$SNV_OPTIONS \
$SV_OPTIONS \
$PARALOG_OPTIONS \
"
# Execute
echo $CMD
bash -c $CMD
```

Optional Settings

Refer to [Command-line Options on page 54](#) for a full list of options.

Somatic UMI Tumor Normal

This recipe is for processing sequencing data with unique molecular identifier (UMI) for somatic tumor normal workflows.

Command Line Example

For Somatic UMI Tumor Normal inputs, tumor and normal sample need to be run separately for the Map/Align stage, and then Variant Calling is started from tumor and normal UMI collapsed BAM.

The following are partial templates that can be used as starting points. Adjust them accordingly for your specific use case.

Map/Align stage

- Configure the INPUT options
- Configure the OUTPUT options

- Configure MAP/ALIGN
- Configure UMI options

```
#!/bin/bash

set -euo pipefail


# Path to DRAGEN hashtable
DRAGEN_HASH_TABLE=<REF_DIR>

# Path to output directory for the DRAGEN run
OUTPUT=<OUT_DIR>

# File prefix for DRAGEN output files
PREFIX=<OUT_PREFIX>

# Define the input sources, select fastq list, fastq, bam, or cram. Please select either tumor or normal input with UMI to generate collapsed BAM. In this example, we use tumor input option.

INPUT_FASTQ_LIST="

--tumor-fastq-list $TUMOR_FASTQ_LIST \
--tumor-fastq-list-sample-id $TUMOR_FASTQ_LIST_SAMPLE_ID \
"

INPUT_FASTQ="

--tumor-fastq1 $TUMOR_FASTQ1 \
--tumor-fastq2 $TUMOR_FASTQ2 \
--RGSM-tumor $RGSM_TUMOR \
"
```

```
--RGID-tumor $RGID_TUMOR \  
"  
  
INPUT_BAM=""  
  
--tumor-bam-input $TUMOR_BAM \  
"  
  
INPUT_CRAM=""  
  
--tumor-cram-input $TUMOR_CRAM \  
"  
  
# Select input source, here in this example we use INPUT_FASTQ_LIST  
  
INPUT_OPTIONS=""  
  
--ref-dir $DRAGEN_HASH_TABLE \  
$INPUT_FASTQ_LIST \  
"  
  
OUTPUT_OPTIONS=""  
  
--output-directory $OUTPUT \  
--output-file-prefix $PREFIX \  
"  
  
MA_OPTIONS=""  
  
--enable-map-align true \  
"
```

```
--enable-sort true \
"
UMI_OPTIONS="

--enable-umi true \
--umi-source $UMI_SOURCE \
--umi-library-type $UMI_LIBRARY_TYPE \
"

# Construct final command line

CMD="

dragen \
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \
$MA_OPTIONS \
$UMI_OPTIONS \
"

#
# Execute

echo $CMD

bash -c $CMD
```

Variant Calling stage

- Configure the INPUT options
- Configure the OUTPUT options
- Configure the VARIANT CALLERs based on the application

- Configure any additional options
- Build up the necessary options for each component separately, so that they can be re-used in the final command line.

```
#!/bin/bash

set -euo pipefail


# Path to DRAGEN hashtable
DRAGEN_HASH_TABLE=<REF_DIR>

# Path to output directory for the DRAGEN run
OUTPUT=<OUT_DIR>

# File prefix for DRAGEN output files
PREFIX=<OUT_PREFIX>

INPUT_BAM="

--tumor-bam-input $TUMOR_BAM \
--bam-input $BAM \
"

INPUT_OPTIONS="

--ref-dir $DRAGEN_HASH_TABLE \
$INPUT_BAM \
"
```

```
OUTPUT_OPTIONS="

--output-directory $OUTPUT \
--output-file-prefix $PREFIX \
"

SNV_OPTIONS="

--enable-variant-caller true \
--vc-enable-umi-solid true or --vc-enable-umi-liquid true \
--vc-target-bed $VC_TARGET_BED \
"

CNV_OPTIONS="

--enable-cnv true \
--cnv-target-bed $CNV_TARGET_BED \
--cnv-normals-list $CNV_PANEL_OF_NORMALS \
"

SV_OPTIONS="

--enable-sv true \
--sv-exome true \
--sv-call-regions-bed $SV_TARGET_BED \
"

# Construct final command line
```

```

CMD="

dragen \
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \
$SNV_OPTIONS \
$CNV_OPTIONS \
$SV_OPTIONS \
"

# Execute
echo $CMD
bash -c $CMD

```

Optional Settings

Optional settings per component are listed below. Refer to [Command-line Options on page 54](#) for a full list of options.

UMI

Option	Description
--umi-source qname/fastq/bamtag	Specify the input type for the UMI sequence. For more information, see UMI Options.
--umi-library-type random-duplex/random-simplex/nonrandom-duplex	Set the batch option for different UMIs correction. For more information, see UMI Options.
--umi-nonrandom-whitelist \$WHITELIST	If UMI is nonrandom, enter the path for a customized, valid UMI sequence.

Option	Description
--umi-min-supporting-reads 2	Specify the number of matching UMI inputs reads required to generate a consensus read. Any family with insufficient supporting reads is discarded. For more information, see UMI Options.
--umi-metrics-interval-file \$UMI_TARGET_BED	Enter the path for target region in BED format.
--umi-emit-multiplicity both	Set the consensus sequence type to output. DRAGEN UMI allows you to collapse duplex sequences from the two strands of the original molecules. For more information, see Merge Duplex UMIs.

SNV

Option	Description
--vc-enable-umi-solid true / --vc-enable-umi-liquid true	When running from UMI data, one of these options is required to let DRAGEN know that the reads have been UMI-collapsed and are therefore more reliable than non-UMI reads. Solid mode is optimized for solid tumors with post collapsed coverage rates of ~200—300X and target allele frequencies of 5% and higher. Liquid mode is optimized for a liquid biopsy pipeline with post collapsed coverage rates of ~2000—2500X and target allele frequencies of 0.4% and higher. As a rough rule of thumb, choose solid for coverage below 1000X and liquid for higher coverage.

Option	Description
<code>--vc-sq-filter-threshold \$THRESHOLD</code>	<p>Threshold for sensitivity-specificity tradeoff. The default threshold is 4(Solid)/2(Liquid). Raise this value to improve specificity at the cost of sensitivity, or lower it to improve sensitivity at the cost of specificity.</p>
<code>--vc-systematic-noise \$SYSTEMATIC_NOISE_FILE</code>	<p>Systematic noise filter. This filter can be useful to remove systematic noise observed in normal samples. Prebuilt systematic noise files are available for download on the Illumina DRAGEN Bio-IT Platform support site page. Alternatively, a systematic noise file can be generated by running the somatic TO pipeline on normal samples. We recommend using a systematic noise file based on normal samples that match the library prep of the tumor samples.</p>
<code>--vc-somatic-hotspots somatic_hotspots_GRCh38.vcf.gz</code>	<p>Hotspots file. By default, DRAGEN treats positions in the COSMIC database as hotspots, assigning an increased prior probability to variants at these positions. Use this option to override with a custom hotspots file if a list of positions of interest is available.</p>

Option	Description
--vc-combine-phased-variants-distance \$DIST	Combining phased variants. By default, DRAGEN will not combine nearby phased calls into MNVs or indels. To combine such calls, set this parameter to a value greater than zero indicating the maximum distance at which calls should be combined. If the user wants to enable the combining of phased variants the recommended value of the distance is 15 base pairs. The valid range is [0; 15]
--vc-enable-liquid-tumor-mode true	Tumor-in-normal contamination. Only use if there is some tumor leakage in the normal control.
--vc-override-tumor-pcr-params-with-normal false	Mixed sample preparation. Only use if the tumor and normal samples exhibit different PCR (indel) noise patterns, eg, due to using different sample preparation.

Option	Description
--vc-target-vaf FLOAT	<p>This option is available with DRAGEN v4.2+.</p> <p>The vc-target-vaf is used to select the variant allele frequencies of interest. The variant caller will aim to detect variants with allele frequencies equal to and larger than this setting. This setting will not apply a hard filter and it is possible to detect variants with allele frequencies lower than the selected threshold. On high coverage and clean datasets, a lower target-vaf may help increase sensitivity. On noisy samples (like FFPE) a higher target-vaf maybe help reduce false positives. Using a low target-vaf may also increase runtime.</p> <p>The valid range is [0, 1]. The default is 0.03 (or 0.001 when --vc-enable-umi-liquid=true).</p>

SNV library specific settings

Option	Description
--vc-target-vaf FLOAT	<p>In FFPE samples with UMI Simplex collapsing it may be beneficial to increase the vc-target-vaf to 0.2 or 0.3. In FFPE samples with UMI Duplex collapsing some of the strand specific FFPE deamination noise may be removed by the duplex collapsing so that the default vc-target-vaf of 0.01 may remain appropriate.</p>

Option	Description
--vc-excluded-regions-bed \$BED	Some FFPE samples may have a high rate of FP calls in SINE (and specifically in ALU) regions. Optionally use an ALU bed to hard filter all calls in this region. Steps are provided below to download an ALU region bed.

Build the SNV systematic noise file

- Run DRAGEN somatic tumor-only on each of approximately 50 normal samples:

```
### choose input either from

### i) BAM

INPUT="--tumor-bam-input ${NORMAL_BAM}"

### ii) FASTQs

INPUT="--tumor-fastq-list ${NORMAL_FASTQ_LIST} \
--tumor-fastq-list-sample-id ${NORMAL_FASTQ_LIST_SAMPLE_ID}"

###


dragen \
-r ${REFERENCE} \
${INPUT} \
--enable-variant-caller=true \
--build-sys-noise-germline-vaf-threshold=1 \
--vc-enable-umi-solid true or --vc-enable-umi-liquid true \
--vc-enable-germline-tagging=true \
--enable-variant-annotation true \
```

```
--variant-annotation-data ${NIRVANA_ANNOTATION_FOLDER} \
--variant-annotation-assembly ${REFERENCE} \
--intermediate-results-dir ${TMP} \
--output-directory ${DIR} \
--output-file-prefix ${PREFIX}
```

2. Gather the full paths to the VCF files from the previous step, and create \${VCF_LIST} by specifying 1 file path per line.
3. Generate the final noise file:

```
dragen \
-r ${REFERENCE} \
--build-sys-noise-vcfs-list ${VCF_LIST} \
--build-sys-noise-method mean \
--intermediate-results-dir ${TMP} \
--output-directory ${DIR} \
--output-file-prefix ${PREFIX}
```

Download a SINE/ALU regions bed for SNV excluded regions

ALUs comprise approximately 11% of the genome and are common in introns. High rates of deamination FP calls have been observed in some FFPE libraries. If the ALU regions are not clinically significant for a specific analysis, use the following steps to filter out the entire ALU region.

1. visit genome.ucsc.edu
2. select the appropriate reference: hg19/hg38
3. specify the following options:
 - Output format: BED
 - Output file name: \${PREFIX}.bed
4. Download the file
5. Open the file and remove redundant headers

6. The 4th column specifies the SINE family. Grep the file using Alu to limit the file to only ALU region
7. Set the excluded regions filter: --vc-excluded-regions-bed \$BED

CNV

Include the matched normal sample in the CNV panel of normals

Option	Description
--cnv-enable-gcbias-correction true	Enable or disable GC bias correction when generating target counts. For more information, see GC Bias Correction on page 186 .
--cnv-segmentation-mode \$SEG_MODE	Specifies the segmentation algorithm to perform. For more information, see Segmentation on page 191 for more information.

Generating Panel of Normals (PON)

Somatic WES CNV requires PON files. The following is an example command of PON generation. Options used for panel of normals generation (BED file, GC Bias Correction, etc) should be matched when processing the case sample. Refer to [Panel of Normals on page 188](#) for more information.

```
CNV_PON_OPTIONS="

--enable-cnv true \

--cnv-target-bed $CNV_TARGET_BED \

"

CMD="

dragen \

$INPUT_OPTIONS \

$OUTPUT_OPTIONS \

$CNV_PON_OPTIONS \

"
"
```

SV

See [Liquid Tumor Calling on page 302](#) for more information.

Option	Description
--sv-enable-liquid-tumor-mode true	Enable liquid tumor mode.
--sv-tin-contam-tolerance \$TIN_CONTAM_TOLERANCE	Set the Tumor-in-Normal (TiN) contamination tolerance level.

Somatic UMI Tumor Only

This recipe is for processing sequencing data with unique molecular identifier (UMI) for somatic tumor only workflows.

Example Command Line

For most scenarios, simply creating the union of the command line options from the single caller scenarios will work.

- Configure the INPUT options
- Configure the OUTPUT options
- Configure MAP/ALIGN depending on if realignment is desired or not
- Configure the VARIANT CALLERs based on the application
- Configure any additional options
- Build up the necessary options for each component separately, so that they can be re-used in the final command line

The following are partial templates that can be used as starting points. Adjust them accordingly for your specific use case.

```
#!/bin/bash

set -euo pipefail


# Path to DRAGEN hashtable
DRAGEN_HASH_TABLE=<REF_DIR>
```

```
# Path to output directory for the DRAGEN run

OUTPUT=<OUT_DIR>

# File prefix for DRAGEN output files

PREFIX=<OUT_PREFIX>

# Define the input sources, select fastq list, fastq, bam, or cram.

INPUT_FASTQ_LIST="

--tumor-fastq-list $TUMOR_FASTQ_LIST \
--tumor-fastq-list-sample-id $TUMOR_FASTQ_LIST_SAMPLE_ID \
"
INPUT_FASTQ="

--tumor-fastq1 $TUMOR_FASTQ1 \
--tumor-fastq2 $TUMOR_FASTQ2 \
--RGSM-tumor $RGSM_TUMOR \
--RGID-tumor $RGID_TUMOR \
"
INPUT_BAM="

--tumor-bam-input $TUMOR_BAM \
--bam-input $BAM \
"
```

```
INPUT_CRAM="

--tumor-cram-input $TUMOR_CRAM \
--cram-input $CRAM \
"

# Select input source, here in this example we use INPUT_FASTQ_LIST

INPUT_OPTIONS="

--ref-dir $DRAGEN_HASH_TABLE \
$INPUT_FASTQ_LIST \
"

OUTPUT_OPTIONS="

--output-directory $OUTPUT \
--output-file-prefix $PREFIX \
"

MA_OPTIONS="

--enable-map-align true \
--enable-sort true \
"

UMI_OPTIONS="

--enable-umi true \
--umi-source $UMI_SOURCE \
"
```

```
--umi-library-type $UMI_LIBRARY_TYPE \
"
SNV_OPTIONS="

--enable-variant-caller true \
--vc-enable-umi-solid true or --vc-enable-umi-liquid true \
--vc-target-bed $VC_TARGET_BED \
--vc-systematic-noise $VC_SYSTEMATIC_NOISE_FILE \
--vc-enable-germline-tagging true \
--enable-variant-annotation true \
--variant-annotation-data $NIRVANA_ANNOTATION_FOLDER \
--variant-annotation-assembly $REFERENCE \
"
CNV_OPTIONS="

--enable-cnv true \
--cnv-target-bed $CNV_TARGET_BED \
--cnv-normals-list $CNV_PANEL_OF_NORMALS \
"
SV_OPTIONS="

--enable-sv true \
--sv-exome true \
--sv-call-regions-bed $SV_TARGET_BED \
```

```

"
# Construct final command line
CMD="

    dragen \
    $INPUT_OPTIONS \
    $OUTPUT_OPTIONS \
    $MA_OPTIONS \
    $UMI_OPTIONS \
    $SNV_OPTIONS \
    $CNV_OPTIONS \
    $SV_OPTIONS \
"

#
# Execute
echo $CMD
bash -c $CMD

```

Optional Settings

Optional settings per component are listed below. Refer to [Command-line Options on page 54](#) for a full list of options.

UMI

Option	Description
--umi-source qname/fastq/bamtag	Specify the input type for the UMI sequence. For more information, see UMI Options.

Option	Description
--umi-library-type random-duplex/random-simplex/nonrandom-duplex	Set the batch option for different UMIs correction. For more information, see UMI Options.
--umi-nonrandom-whitelist \$WHITELIST	If UMI is nonrandom, enter the path for a customized, valid UMI sequence.
--umi-min-supporting-reads 2	Specify the number of matching UMI inputs reads required to generate a consensus read. Any family with insufficient supporting reads is discarded. For more information, see UMI Options.
--umi-metrics-interval-file \$UMI_TARGET_BED	Enter the path for target region in BED format.
--umi-emit-multiplicity both	Set the consensus sequence type to output. DRAGEN UMI allows you to collapse duplex sequences from the two strands of the original molecules. For more information, see Merge Duplex UMIs.

SNV

Option	Description
--vc-enable-umi-solid true / --vc-enable-umi-liquid true	When running from UMI data, one of these options is required to let DRAGEN know that the reads have been UMI-collapsed and are therefore more reliable than non-UMI reads. Solid mode is optimized for solid tumors with post collapsed coverage rates of ~200—300X and target allele frequencies of 5% and higher. Liquid mode is optimized for a liquid biopsy pipeline with post collapsed coverage rates of ~2000—2500X and target allele frequencies of 0.4% and higher. As a rough rule of thumb, choose solid for coverage below 1000X and liquid for higher coverage.
--vc-sq-filter-threshold \$THRESHOLD	Threshold for sensitivity-specificity tradeoff. The default threshold is 4(Solid)/2(Liquid). Raise this value to improve specificity at the cost of sensitivity, or lower it to improve sensitivity at the cost of specificity.

Option	Description
--vc-systematic-noise \$SYSTEMATIC_NOISE_FILE	Systematic noise filter. This filter can be useful to remove systematic noise observed in normal samples. Prebuilt systematic noise files are available for download on the Illumina DRAGEN Bio-IT Platform support site page. Alternatively, a systematic noise file can be generated by running the somatic TO pipeline on normal samples. We recommend using a systematic noise file based on normal samples that match the library prep of the tumor samples.
--vc-somatic-hotspots somatic_hotspots_GRCh38.vcf.gz	Hotspots file. By default, DRAGEN treats positions in the COSMIC database as hotspots, assigning an increased prior probability to variants at these positions. Use this option to override with a custom hotspots file if a list of positions of interest is available.
--vc-combine-phased-variants-distance \$DIST	Combining phased variants. By default, DRAGEN will not combine nearby phased calls into MNVs or indels. To combine such calls, set this parameter to a value greater than zero indicating the maximum distance at which calls should be combined. If the user wants to enable the combining of phased variants the recommended value of the distance is 15 base pairs. The valid range is [0; 15]

Option	Description
<pre>--vc-enable-germline-tagging true --enable-variant-annotation true --variant-annotation-data \$NIRVANA_ANNOTATION_FOLDER --variant-annotation-assembly \$REFERENCE</pre>	Germline filtering. Enable to tag variants as germline or somatic based on population databases. \$REFERENCE can be GRCh37 or GRCh38 (GRCh37 is compatible with hs37d5 and hg19). The Nirvana annotation database is downloadable at this page.
<pre>--vc-target-vaf FLOAT</pre>	<p>This option is available with DRAGEN v4.2+.</p> <p>The vc-target-vaf is used to select the variant allele frequencies of interest. The variant caller will aim to detect variants with allele frequencies equal to and larger than this setting. This setting will not apply a hard filter and it is possible to detect variants with allele frequencies lower than the selected threshold. On high coverage and clean datasets, a lower target-vaf may help increase sensitivity. On noisy samples (like FFPE) a higher target-vaf maybe help reduce false positives. Using a low target-vaf may also increase runtime.</p> <p>The valid range is [0, 1]. The default is 0.03 (or 0.001 when --vc-enable-umi-liquid=true).</p>

SNV library specific settings

Option	Description
--vc-target-vaf FLOAT	In FFPE samples with UMI Simplex collapsing it may be beneficial to increase the vc-target-vaf to 0.2 or 0.3. In FFPE samples with UMI Duplex collapsing some of the strand specific FFPE deamination noise may be removed by the duplex collapsing so that the default vc-target-vaf of 0.01 may remain appropriate.
--vc-excluded-regions-bed \$BED	Some FFPE samples may have a high rate of FP calls in SINE (and specifically in ALU) regions. Optionally use an ALU bed to hard filter all calls in this region. Steps are provided below to download an ALU region bed.

Build the SNV systematic noise file

- Run DRAGEN somatic tumor-only on each of approximately 50 normal samples:

```
### choose input either from
### i) BAM
INPUT="--tumor-bam-input ${NORMAL_BAM}"
### ii) FASTQs
INPUT="--tumor-fastq-list ${NORMAL_FASTQ_LIST} \
--tumor-fastq-list-sample-id ${NORMAL_FASTQ_LIST_SAMPLE_ID}"
###

dragen \
```

```

-r ${REFERENCE} \
${INPUT} \
--enable-variant-caller=true \
--vc-enable-umi-solid true or --vc-enable-umi-liquid true \
--build-sys-noise-germline-vaf-threshold=1 \
--vc-enable-germline-tagging=true \
--enable-variant-annotation true \
--variant-annotation-data ${NIRVANA_ANNOTATION_FOLDER} \
--variant-annotation-assembly ${REFERENCE} \
--intermediate-results-dir ${TMP} \
--output-directory ${DIR} \
--output-file-prefix ${PREFIX}

```

2. Gather the full paths to the VCF files from the previous step, and create \${VCF_LIST} by specifying 1 file path per line.
3. Generate the final noise file:

```

dragen \
-r ${REFERENCE} \
--build-sys-noise-vcfs-list ${VCF_LIST} \
--build-sys-noise-method mean \
--intermediate-results-dir ${TMP} \
--output-directory ${DIR} \
--output-file-prefix ${PREFIX}

```

Download a SINE/ALU regions bed for SNV excluded regions

ALUs comprise approximately 11% of the genome and are common in introns. High rates of deamination FP calls have been observed in some FFPE libraries. If the ALU regions are not clinically significant for a specific analysis, use the following steps to filter out the entire ALU region.

1. visit genome.ucsc.edu
2. select the appropriate reference: hg19/hg38
3. specify the following options:
 - Output format: BED
 - Output file name: \${PREFIX}.bed
4. Download the file
5. Open the file and remove redundant headers
6. The 4th column specifies the SINE family. Grep the file using Alu to limit the file to only ALU region
7. Set the excluded regions filter: --vc-excluded-regions-bed \$BED

CNV

Include the matched normal sample in the CNV panel of normals

Option	Description
--cnv-enable-gcbias-correction true	Enable or disable GC bias correction when generating target counts. For more information, see GC Bias Correction on page 186 .
--cnv-segmentation-mode \$SEG_MODE	Specifies the segmentation algorithm to perform. For more information, see Segmentation on page 191 for more information.

Generating Panel of Normals (PON)

Somatic WES CNV requires PON files. The following is an example command of PON generation. Options used for panel of normals generation (BED file, GC Bias Correction, etc) should be matched when processing the case sample. Refer to [Panel of Normals on page 188](#) for more information.

```
CNV_PON_OPTIONS="

--enable-cnv true \
```

```
--cnv-target-bed $CNV_TARGET_BED \
"
CMD="

dragen \
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \
$CNV_PON_OPTIONS \
"
"
```

SV

See [Liquid Tumor Calling on page 302](#)for more information.

Option	Description
--sv-systematic-noise \$SYSTEMATIC_NOISE_BEDPE	Systematic noise BEDPE file containing the set of noisy paired regions (optionally gzip or bzip compressed).

Somatic WES Tumor Normal

This recipe is for processing whole exome sequencing data for somatic tumor normal workflows.

Example Command Line

For most scenarios, simply creating the union of the command line options from the single caller scenarios will work.

- Configure the INPUT options
- Configure the OUTPUT options
- Configure MAP/ALIGN depending on if realignment is desired or not
- Configure the VARIANT CALLERs based on the application
- Configure any additional options

- Build up the necessary options for each component separately, so that they can be re-used in the final command line

The following are partial templates that can be used as starting points. Adjust them accordingly for your specific use case:

```
#!/bin/bash

set -euo pipefail


# Path to DRAGEN hashtable
DRAGEN_HASH_TABLE=<REF_DIR>

# Path to output directory for the DRAGEN run
OUTPUT=<OUT_DIR>

# File prefix for DRAGEN output files
PREFIX=<OUT_PREFIX>

# Define the input sources, select fastq list, fastq, bam, or cram.
INPUT_FASTQ_LIST="

--tumor-fastq-list $TUMOR_FASTQ_LIST \
--tumor-fastq-list-sample-id $TUMOR_FASTQ_LIST_SAMPLE_ID \
--fastq-list $FASTQ_LIST \
--fastq-list-sample-id $FASTQ_LIST_SAMPLE_ID \
"

INPUT_FASTQ="
```

```
--tumor-fastq1 $TUMOR_FASTQ1 \
--tumor-fastq2 $TUMOR_FASTQ2 \
--RGSM-tumor $RGSM_TUMOR \
--RGID-tumor $RGID_TUMOR \
--fastq-file1 $FASTQ1 \
--fastq-file2 $FASTQ2 \
--RGSM $RGSM \
--RGID $RGID \
"
"

INPUT_BAM="

--tumor-bam-input $TUMOR_BAM \
--bam-input $BAM \
"
"

INPUT_CRAM="

--tumor-cram-input $TUMOR_CRAM \
--cram-input $CRAM \
"
"

# Select input source, here in this example we use INPUT_FASTQ_LIST

INPUT_OPTIONS="

--ref-dir $DRAGEN_HASH_TABLE \
$INPUT_FASTQ_LIST \
"
```

```
"  
  
OUTPUT_OPTIONS=""  
  
    --output-directory $OUTPUT \  
    --output-file-prefix $PREFIX \  
  
"  
  
  
  
MA_OPTIONS=""  
  
    --enable-map-align true \  
    --enable-sort true \  
    --enable-duplicate-marking true \  
  
"  
  
  
  
CNV_OPTIONS=""  
  
    --enable-cnv true \  
    --cnv-target-bed $CNV_TARGET_BED \  
    --cnv-normals-list $CNV_PANEL_OF_NORMALS \  
    --cnv-use-somatic-vc-baf true \  
  
"  
  
  
  
SNV_OPTIONS=""  
  
    --enable-variant-caller true \  
    --vc-target-bed $VC_TARGET_BED \  
  
"
```

```
SV_OPTIONS="

--enable-sv true \
--sv-exome true \
--sv-call-regions-bed $SV_TARGET_BED \
"

SNV_SV_DEDUPLICATION_OPTIONS="

--enable-variant-deduplication true \
"

# Construct final command line

CMD="

dragen \
$INPUT_OPTIONS \
$output_options \
$MA_OPTIONS \
$CNV_OPTIONS \
$SNV_OPTIONS \
$SV_OPTIONS \
$SNV_SV_DEDUPLICATION_OPTIONS \
"

# Execute
```

```
echo $CMD
bash -c $CMD
```

Optional Settings

Optional settings per component are listed below. Refer to [Command-line Options on page 54](#) for a full list of options.

CNV

Include the matched normal sample in the CNV panel of normals

Option	Description
--cnv-enable-gcbias-correction true	Enable or disable GC bias correction when generating target counts. For more information, see GC Bias Correction on page 186 .
--cnv-segmentation-mode \$SEG_MODE	Specifies the segmentation algorithm to perform. For more information, see Segmentation on page 191 for more information.

Generating Panel of Normals (PON)

Somatic WES CNV requires PON files. The following is an example command of PON generation. Options used for panel of normals generation (BED file, GC Bias Correction, etc) should be matched when processing the case sample. Refer to [Panel of Normals on page 188](#) for more information.

```
CNV_PON_OPTIONS="
--enable-cnv true \
--cnv-target-bed $CNV_TARGET_BED \
"
CMD=""
```

```
dragen \
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \
$CNV_PON_OPTIONS \
"
```

SNV

Option	Description
--vc-sq-filter-threshold \$THRESHOLD	Threshold for sensitivity-specificity tradeoff. The default threshold is 4(Solid)/2(Liquid). Raise this value to improve specificity at the cost of sensitivity, or lower it to improve sensitivity at the cost of specificity.
--vc-systematic-noise \$SYSTEMATIC_NOISE_FILE	Systematic noise filter. This filter can be useful to remove systematic noise observed in normal samples. Prebuilt systematic noise files are available for download on the Illumina DRAGEN Bio-IT Platform support site page. Alternatively, a systematic noise file can be generated by running the somatic TO pipeline on normal samples. We recommend using a systematic noise file based on normal samples that match the library prep of the tumor samples.

Option	Description
--vc-somatic-hotspots somatic_hotspots_ GRCh38.vcf.gz	Hotspots file. By default, DRAGEN treats positions in the COSMIC database as hotspots, assigning an increased prior probability to variants at these positions. Use this option to override with a custom hotspots file if a list of positions of interest is available.
--vc-combine-phased-variants-distance \$DIST	Combining phased variants. By default, DRAGEN will not combine nearby phased calls into MNVs or indels. To combine such calls, set this parameter to a value greater than zero indicating the maximum distance at which calls should be combined. If the user wants to enable the combining of phased variants the recommended value of the distance is 15 base pairs. The valid range is [0; 15]
--vc-enable-liquid-tumor-mode true	Tumor-in-normal contamination. Only use if there is some tumor leakage in the normal control.
--vc-override-tumor-pcr-params-with-normal false	Mixed sample preparation. Only use if the tumor and normal samples exhibit different PCR (indel) noise patterns, eg, due to using different sample preparation.

Option	Description
--vc-target-vaf FLOAT	<p>This option is available with DRAGEN v4.2+.</p> <p>The vc-target-vaf is used to select the variant allele frequencies of interest. The variant caller will aim to detect variants with allele frequencies equal to and larger than this setting. This setting will not apply a hard filter and it is possible to detect variants with allele frequencies lower than the selected threshold. On high coverage and clean datasets, a lower target-vaf may help increase sensitivity. On noisy samples (like FFPE) a higher target-vaf maybe help reduce false positives. Using a low target-vaf may also increase runtime.</p> <p>The valid range is [0, 1]. The default is 0.03 (or 0.001 when --vc-enable-umi-liquid=true).</p>

SNV library specific settings

Option	Description
--vc-target-vaf FLOAT	<p>In FFPE samples with UMI Simplex collapsing it may be beneficial to increase the vc-target-vaf to 0.2 or 0.3. In FFPE samples with UMI Duplex collapsing some of the strand specific FFPE deamination noise may be removed by the duplex collapsing so that the default vc-target-vaf of 0.01 may remain appropriate.</p>

Option	Description
--vc-excluded-regions-bed \$BED	Some FFPE samples may have a high rate of FP calls in SINE (and specifically in ALU) regions. Optionally use an ALU bed to hard filter all calls in this region. Steps are provided below to download an ALU region bed.

Build the SNV systematic noise file

- Run DRAGEN somatic tumor-only on each of approximately 50 normal samples:

```
### choose input either from

### i) BAM

INPUT="--tumor-bam-input ${NORMAL_BAM}"

### ii) FASTQs

INPUT="--tumor-fastq-list ${NORMAL_FASTQ_LIST} \
--tumor-fastq-list-sample-id ${NORMAL_FASTQ_LIST_SAMPLE_ID}"

###


dragen \
-r ${REFERENCE} \
${INPUT} \
--enable-variant-caller=true \
--vc-detect-systematic-noise true \
--build-sys-noise-germline-vaf-threshold=1 \
--vc-enable-germline-tagging true \
--enable-variant-annotation true \
```

```
--variant-annotation-data ${NIRVANA_ANNOTATION_FOLDER} \
--variant-annotation-assembly ${REFERENCE} \
--intermediate-results-dir ${TMP} \
--output-directory ${DIR} \
--output-file-prefix ${PREFIX}
```

2. Gather the full paths to the VCF files from the previous step, and create \${VCF_LIST} by specifying 1 file path per line.
3. Generate the final noise file

```
dragen \
-r ${REFERENCE} \
--build-sys-noise-vcfs-list ${VCF_LIST} \
--build-sys-noise-method mean \
--intermediate-results-dir ${TMP} \
--output-directory ${DIR} \
--output-file-prefix ${PREFIX}
```

Download a SINE/ALU regions bed for SNV excluded regions

ALUs comprise approximately 11% of the genome and are common in introns. High rates of deamination FP calls have been observed in some FFPE libraries. If the ALU regions are not clinically significant for a specific analysis, use the following steps to filter out the entire ALU region.

1. visit genome.ucsc.edu
2. select the appropriate reference: hg19/hg38
3. specify the following options:
 - Output format: BED
 - Output file name: \${PREFIX}.bed
4. Download the file
5. Open the file and remove redundant headers

6. The 4th column specifies the SINE family. Grep the file using Alu to limit the file to only ALU region
7. Set the excluded regions filter: --vc-excluded-regions-bed \$BED

SV

See [Liquid Tumor Calling on page 302](#) for more information.

Option	Description
--sv-enable-liquid-tumor-mode true	Enable liquid tumor mode.
--sv-tin-contam-tolerance \$TIN_CONTAM_TOLERANCE	Set the Tumor-in-Normal (TiN) contamination tolerance level.

SNV-SV Deduplication

Incorporating both SV and SNV callers in somatic workflows can increase sensitivity and prevent the occurrence of replicated variants within genes such as FLT3 and KMT2A. --enable-variant-deduplication true should be used to filter all small indels in the structural variant VCF that appear and are passing in the small variant VCF, PASS in the FILTER column of the small variant VCF file. DRAGEN normalizes variants by trimming and left shifting up to 500 bases. The new deduplicated SV VCF file will have the same prefix passed by --output-file-prefix followed by sv.small_indel_dedup.

Somatic WES Tumor Only

This recipe is for processing whole exome sequencing data for somatic tumor only workflows.

Example Command Line

For most scenarios, simply creating the union of the command line options from the single caller scenarios will work.

- Configure the INPUT options
- Configure the OUTPUT options
- Configure MAP/ALIGN depending on if realignment is desired or not
- Configure the VARIANT CALLERs based on the application
- Configure any additional options
- Build up the necessary options for each component separately, so that they can be re-used in the final command line

The following are partial templates that can be used as starting points. Adjust them accordingly for your specific use case.

```
#!/bin/bash

set -euo pipefail


# Path to DRAGEN hashtable
DRAGEN_HASH_TABLE=<REF_DIR>

# Path to output directory for the DRAGEN run
OUTPUT=<OUT_DIR>

# File prefix for DRAGEN output files
PREFIX=<OUT_PREFIX>

# Define the input sources, select fastq list, fastq, bam, or cram.
INPUT_FASTQ_LIST="

--tumor-fastq-list $TUMOR_FASTQ_LIST \
--tumor-fastq-list-sample-id $TUMOR_FASTQ_LIST_SAMPLE_ID \
"
INPUT_FASTQ="

--tumor-fastq1 $TUMOR_FASTQ1 \
--tumor-fastq2 $TUMOR_FASTQ2 \
--RGSM-tumor $RGSM_TUMOR \
--RGID-tumor $RGID_TUMOR \
"
"
```

```
INPUT_BAM="

--tumor-bam-input $TUMOR_BAM \
--bam-input $BAM \
"

INPUT_CRAM="

--tumor-cram-input $TUMOR_CRAM \
--cram-input $CRAM \
"

# Select input source, here in this example we use INPUT_FASTQ_LIST

INPUT_OPTIONS="

--ref-dir $DRAGEN_HASH_TABLE \
$INPUT_FASTQ_LIST \
"

OUTPUT_OPTIONS="

--output-directory $OUTPUT \
--output-file-prefix $PREFIX \
"

MA_OPTIONS="

--enable-map-align true \
"
```

```
--enable-sort true \
--enable-duplicate-marking true \
"
CNV_OPTIONS="

--enable-cnv true \
--cnv-target-bed $CNV_TARGET_BED \
--cnv-normals-list $CNV_PANEL_OF_NORMALS \
"
SNV_OPTIONS="

--enable-variant-caller true \
--vc-target-bed $VC_TARGET_BED \
--vc-systematic-noise $VC_SYSTEMATIC_NOISE_FILE \
--vc-enable-germline-tagging true \
--enable-variant-annotation true \
--variant-annotation-data $NIRVANA_ANNOTATION_FOLDER \
--variant-annotation-assembly $REFERENCE \
"
SV_OPTIONS="

--enable-sv true \
--sv-exome true \
--sv-call-regions-bed $SV_TARGET_BED \
```

```
"  
  
SNV_SV_DEDUPLICATION_OPTIONS="  
  
    --enable-variant-deduplication true \  
  
"  
  
# Construct final command line  
  
CMD="  
  
    dragen \  
  
    $INPUT_OPTIONS \  
  
    $OUTPUT_OPTIONS \  
  
    $MA_OPTIONS \  
  
    $CNV_OPTIONS \  
  
    $SNV_OPTIONS \  
  
    $SV_OPTIONS \  
  
    $SNV_SV_DEDUPLICATION_OPTIONS \  
  
"  
  
# Execute  
  
echo $CMD  
  
bash -c $CMD
```

Optional Settings

Optional settings per component are listed below. Refer to [Command-line Options on page 54](#) for a full list of options.

CNV

Include the matched normal sample in the CNV panel of normals

Option	Description
--cnv-enable-gcbias-correction true	Enable or disable GC bias correction when generating target counts. For more information, see GC Bias Correction on page 186 .
--cnv-segmentation-mode \$SEG_MODE	Specifies the segmentation algorithm to perform. For more information, see Segmentation on page 191 for more information.

Generating Panel of Normals (PON)

Somatic WES CNV requires PON files. The following is an example command of PON generation. Options used for panel of normals generation (BED file, GC Bias Correction, etc) should be matched when processing the case sample. Refer to [Panel of Normals on page 188](#) for more information.

```
CNV_PON_OPTIONS="

--enable-cnv true \

--cnv-target-bed $CNV_TARGET_BED \

"

CMD="

dragen \

$INPUT_OPTIONS \

$OUTPUT_OPTIONS \

$CNV_PON_OPTIONS \

"
"
```

SNV

Option	Description
--vc-sq-filter-threshold \$THRESHOLD	Threshold for sensitivity-specificity tradeoff. The default threshold is 4(Solid)/2(Liquid). Raise this value to improve specificity at the cost of sensitivity, or lower it to improve sensitivity at the cost of specificity.
--vc-systematic-noise \$SYSTEMATIC_NOISE_FILE	Systematic noise filter. This filter can be useful to remove systematic noise observed in normal samples. Prebuilt systematic noise files are available for download on the Illumina DRAGEN Bio-IT Platform support site page. Alternatively, a systematic noise file can be generated by running the somatic TO pipeline on normal samples. We recommend using a systematic noise file based on normal samples that match the library prep of the tumor samples.
--vc-somatic-hotspots somatic_hotspots_GRCh38.vcf.gz	Hotspots file. By default, DRAGEN treats positions in the COSMIC database as hotspots, assigning an increased prior probability to variants at these positions. Use this option to override with a custom hotspots file if a list of positions of interest is available.

Option	Description
<code>--vc-combine-phased-variants-distance \$DIST</code>	Combining phased variants. By default, DRAGEN will not combine nearby phased calls into MNVs or indels. To combine such calls, set this parameter to a value greater than zero indicating the maximum distance at which calls should be combined. If the user wants to enable the combining of phased variants the recommended value of the distance is 15 base pairs. The valid range is [0; 15]
<code>--vc-enable-germline-tagging true --enable-variant-annotation true --variant-annotation-data \$NIRVANA_ANNOTATION_FOLDER --variant-annotation-assembly \$REFERENCE</code>	Germline filtering. Enable to tag variants as germline or somatic based on population databases. \$REFERENCE can be GRCh37 or GRCh38 (GRCh37 is compatible with hs37d5 and hg19). The Nirvana annotation database is downloadable at this page.

Option	Description
--vc-target-vaf FLOAT	This option is available with DRAGEN v4.2+. The vc-target-vaf is used to select the variant allele frequencies of interest. The variant caller will aim to detect variants with allele frequencies equal to and larger than this setting. This setting will not apply a hard filter and it is possible to detect variants with allele frequencies lower than the selected threshold. On high coverage and clean datasets, a lower target-vaf may help increase sensitivity. On noisy samples (like FFPE) a higher target-vaf maybe help reduce false positives. Using a low target-vaf may also increase runtime. The valid range is [0, 1]. The default is 0.03 (or 0.001 when --vc-enable-umi-liquid=true).

SNV library specific settings

Option	Description
--vc-target-vaf FLOAT	In FFPE samples with UMI Simplex collapsing it may be beneficial to increase the vc-target-vaf to 0.2 or 0.3. In FFPE samples with UMI Duplex collapsing some of the strand specific FFPE deamination noise may be removed by the duplex collapsing so that the default vc-target-vaf of 0.01 may remain appropriate.

Option	Description
--vc-excluded-regions-bed \$BED	Some FFPE samples may have a high rate of FP calls in SINE (and specifically in ALU) regions. Optionally use an ALU bed to hard filter all calls in this region. Steps are provided below to download an ALU region bed.

Build the SNV systematic noise file

- Run DRAGEN somatic tumor-only on each of approximately 50 normal samples:

```
### choose input either from

### i) BAM

INPUT="--tumor-bam-input ${NORMAL_BAM}"

### ii) FASTQs

INPUT="--tumor-fastq-list ${NORMAL_FASTQ_LIST} \
--tumor-fastq-list-sample-id ${NORMAL_FASTQ_LIST_SAMPLE_ID}"

###


dragen \
-r ${REFERENCE} \
${INPUT} \
--enable-variant-caller true \
--vc-detect-systematic-noise true \
--build-sys-noise-germline-vaf-threshold=1 \
--vc-enable-germline-tagging true \
--enable-variant-annotation true \
```

```
--variant-annotation-data ${NIRVANA_ANNOTATION_FOLDER} \
--variant-annotation-assembly ${REFERENCE} \
--intermediate-results-dir ${TMP} \
--output-directory ${DIR} \
--output-file-prefix ${PREFIX}
```

2. Gather the full paths to the VCF files from the previous step, and create \${VCF_LIST} by specifying 1 file path per line.
3. Generate the final noise file

```
dragen \
-r ${REFERENCE} \
--build-sys-noise-vcfs-list ${VCF_LIST} \
--build-sys-noise-method mean \
--intermediate-results-dir ${TMP} \
--output-directory ${DIR} \
--output-file-prefix ${PREFIX}
```

Download a SINE/ALU regions bed for SNV excluded regions

ALUs comprise approximately 11% of the genome and are common in introns. High rates of deamination FP calls have been observed in some FFPE libraries. If the ALU regions are not clinically significant for a specific analysis, use the following steps to filter out the entire ALU region.

1. visit genome.ucsc.edu
2. select the appropriate reference: hg19/hg38
3. specify the following options:
 - Output format: BED
 - Output file name: \${PREFIX}.bed
4. Download the file
5. Open the file and remove redundant headers

6. The 4th column specifies the SINE family. Grep the file using Alu to limit the file to only ALU region
7. Set the excluded regions filter: --vc-excluded-regions-bed \$BED

SNV-SV Deduplication

Incorporating both SV and SNV callers in somatic workflows can increase sensitivity and prevent the occurrence of replicated variants within genes such as FLT3 and KMT2A. --enable-variant-deduplication true should be used to filter all small indels in the structural variant VCF that appear and are passing in the small variant VCF, PASS in the FILTER column of the small variant VCF file. DRAGEN normalizes variants by trimming and left shifting up to 500 bases. The new deduplicated SV VCF file will have the same prefix passed by --output-file-prefix followed by sv.small_indel_dedup.

Somatic WGS Tumor Normal

This recipe is for processing whole genome sequencing data for somatic tumor normal workflows.

Example Command Line

For most scenarios, simply creating the union of the command line options from the single caller scenarios will work.

- Configure the INPUT options
- Configure the OUTPUT options
- Configure MAP/ALIGN depending on if realignment is desired or not
- Configure the VARIANT CALLERs based on the application
- Configure any additional options
- Build up the necessary options for each component separately, so that they can be re-used in the final command line

The following are partial templates that can be used as starting points. Adjust them accordingly for your specific use case.

```
#!/bin/bash

set -euo pipefail


# Path to DRAGEN hashtable

DRAGEN_HASH_TABLE=<REF_DIR>
```

```
# Path to output directory for the DRAGEN run
OUTPUT=<OUT_DIR>

# File prefix for DRAGEN output files
PREFIX=<OUT_PREFIX>

# Define the input sources, select fastq list, fastq, bam, or cram.
INPUT_FASTQ_LIST="

--tumor-fastq-list $TUMOR_FASTQ_LIST \
--tumor-fastq-list-sample-id $TUMOR_FASTQ_LIST_SAMPLE_ID \
--fastq-list $FASTQ_LIST \
--fastq-list-sample-id $FASTQ_LIST_SAMPLE_ID \
"

INPUT_FASTQ="

--tumor-fastq1 $TUMOR_FASTQ1 \
--tumor-fastq2 $TUMOR_FASTQ2 \
--RGSM-tumor $RGSM_TUMOR \
--RGID-tumor $RGID_TUMOR \
--fastq-file1 $FASTQ1 \
--fastq-file2 $FASTQ2 \
--RGSM $RGSM \
--RGID $RGID \
```

```
"  
  
INPUT_BAM=""  
  
--tumor-bam-input $TUMOR_BAM \  
  
--bam-input $BAM \  
  
"  
  
  
  
INPUT_CRAM=""  
  
--tumor-cram-input $TUMOR_CRAM \  
  
--cram-input $CRAM \  
  
"  
  
  
  
# Select input source, here in this example we use INPUT_FASTQ_LIST  
  
INPUT_OPTIONS=""  
  
--ref-dir $DRAGEN_HASH_TABLE \  
  
$INPUT_FASTQ_LIST \  
  
"  
  
  
  
OUTPUT_OPTIONS=""  
  
--output-directory $OUTPUT \  
  
--output-file-prefix $PREFIX \  
  
"  
  
  
  
MA_OPTIONS=""
```

```
--enable-map-align true \
--enable-sort true \
--enable-duplicate-marking true \
"
CNV_OPTIONS="

--enable-cnv true \
--cnv-use-somatic-vc-baf true \
"
SNV_OPTIONS="

--enable-variant-caller true \
"
SV_OPTIONS="

--enable-sv true \
"
SNV_SV_DEDUPPLICATION_OPTIONS="

--enable-variant-deduplication true \
"
#
# Construct final command line
CMD="
```

```

dragen \
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \
$MA_OPTIONS \
$CNV_OPTIONS \
$SNV_OPTIONS \
$SV_OPTIONS \
$SNV_SV_DEDUPLICATION_OPTIONS \
"
# Execute
echo $CMD
bash -c $CMD

```

Optional Settings

Optional settings per component are listed below. Refer to [Command-line Options on page 54](#) for a full list of options.

CNV

Option	Description
--cnv-filter-length \$CNV_FILTER_LENGTH	Specifies the minimum event length in bases at which a reported event is marked as PASS in the output VCF file.
--cnv-merge-distance \$CNV_MERGE_DISTANCE	Specifies the maximum segment gap allowed for merging segments.

Option	Description
--cnv-normal-cnv-vcf \$CNV_NORMAL_VCF	Specify germline CNVs from the matched normal sample. For more information, see Germline-Aware Mode on page 222 .
--cnv-somatic-enable-het-calling true	Enable HET-calling mode for heterogeneous segments. For more information, see HET-calling Mode on page 223 .

SNV

Option	Description
--vc-sq-filter-threshold \$THRESHOLD	Threshold for sensitivity-specificity tradeoff. The default threshold is 4(Solid)/2(Liquid). Raise this value to improve specificity at the cost of sensitivity, or lower it to improve sensitivity at the cost of specificity.
--vc-systematic-noise \$SYSTEMATIC_NOISE_FILE	Systematic noise filter. This filter can be useful to remove systematic noise observed in normal samples. Prebuilt systematic noise files are available for download on the Illumina DRAGEN Bio-IT Platform support site page. Alternatively, a systematic noise file can be generated by running the somatic TO pipeline on normal samples. We recommend using a systematic noise file based on normal samples that match the library prep of the tumor samples.

Option	Description
--vc-somatic-hotspots somatic_hotspots_ GRCh38.vcf.gz	Hotspots file. By default, DRAGEN treats positions in the COSMIC database as hotspots, assigning an increased prior probability to variants at these positions. Use this option to override with a custom hotspots file if a list of positions of interest is available.
--vc-combine-phased-variants-distance \$DIST	Combining phased variants. By default, DRAGEN will not combine nearby phased calls into MNVs or indels. To combine such calls, set this parameter to a value greater than zero indicating the maximum distance at which calls should be combined. If the user wants to enable the combining of phased variants the recommended value of the distance is 15 base pairs. The valid range is [0; 15]
--vc-enable-liquid-tumor-mode true	Tumor-in-normal contamination. Only use if there is some tumor leakage in the normal control.
--vc-override-tumor-pcr-params-with-normal false	Mixed sample preparation. Only use if the tumor and normal samples exhibit different PCR (indel) noise patterns, eg, due to using different sample preparation.

Option	Description
--vc-target-vaf FLOAT	<p>This option is available with DRAGEN v4.2+.</p> <p>The vc-target-vaf is used to select the variant allele frequencies of interest. The variant caller will aim to detect variants with allele frequencies equal to and larger than this setting. This setting will not apply a hard filter and it is possible to detect variants with allele frequencies lower than the selected threshold. On high coverage and clean datasets, a lower target-vaf may help increase sensitivity. On noisy samples (like FFPE) a higher target-vaf maybe help reduce false positives. Using a low target-vaf may also increase run time.</p> <p>The valid range is [0, 1]. The default is 0.03 (or 0.001 when --vc-enable-umi-liquid=true).</p>

SNV library specific settings

Option	Description
--vc-target-vaf FLOAT	<p>In FFPE samples with UMI Simplex collapsing it may be beneficial to increase the vc-target-vaf to 0.2 or 0.3. In FFPE samples with UMI Duplex collapsing some of the strand specific FFPE deamination noise may be removed by the duplex collapsing so that the default vc-target-vaf of 0.01 may remain appropriate.</p>

Option	Description
--vc-excluded-regions-bed \$BED	Some FFPE samples may have a high rate of FP calls in SINE (and specifically in ALU) regions. Optionally use an ALU bed to hard filter all calls in this region. Steps are provided below to download an ALU region bed.

Build the SNV systematic noise file

- Run DRAGEN somatic tumor-only on each of approximately 50 normal samples

```
### choose input either from

### i) BAM

INPUT="--tumor-bam-input ${NORMAL_BAM}"

### ii) FASTQs

INPUT="--tumor-fastq-list ${NORMAL_FASTQ_LIST} \
--tumor-fastq-list-sample-id ${NORMAL_FASTQ_LIST_SAMPLE_ID}"

###


dragen \
-r ${REFERENCE} \
${INPUT} \
--enable-variant-caller true \
--vc-detect-systematic-noise true \
--build-sys-noise-germline-vaf-threshold=1 \
--vc-enable-germline-tagging true \
--enable-variant-annotation true \
```

```
--variant-annotation-data ${NIRVANA_ANNOTATION_FOLDER} \
--variant-annotation-assembly ${REFERENCE} \
--intermediate-results-dir ${TMP} \
--output-directory ${DIR} \
--output-file-prefix ${PREFIX}
```

2. Gather the full paths to the VCF files from the previous step, and create \${VCF_LIST} by specifying 1 file path per line.
3. Generate the final noise file

```
dragen \
-r ${REFERENCE} \
--build-sys-noise-vcfs-list ${VCF_LIST} \
--build-sys-noise-method max \
--intermediate-results-dir ${TMP} \
--output-directory ${DIR} \
--output-file-prefix ${PREFIX}
```

Download a SINE/ALU regions bed for SNV excluded regions

ALUs comprise approximately 11% of the genome and are common in introns. High rates of deamination FP calls have been observed in some FFPE libraries. If the ALU regions are not clinically significant for a specific analysis, use the following steps to filter out the entire ALU region.

1. visit genome.ucsc.edu
2. select the appropriate reference: hg19/hg38
3. specify the following options:
 - Output format: BED
 - Output file name: \${PREFIX}.bed
4. Download the file
5. Open the file and remove redundant headers

6. The 4th column specifies the SINE family. Grep the file using Alu to limit the file to only ALU region
7. Set the excluded regions filter: --vc-excluded-regions-bed \$BED

SV

See [Liquid Tumor Calling on page 302](#) for more information.

Option	Description
--sv-enable-liquid-tumor-mode true	Enable liquid tumor mode.
--sv-tin-contam-tolerance \$TIN_CONTAM_TOLERANCE	Set the Tumor-in-Normal (TiN) contamination tolerance level.
--sv-systematic-noise \$SYSTEMATIC_NOISE_BEDPE	Systematic noise BEDPE file containing the set of noisy paired regions (optionally gzip or bzip compressed). For more information, see Systematic Noise Filtering on page 137 .

SNV-SV Deduplication

Incorporating both SV and SNV callers in somatic workflows can increase sensitivity and prevent the occurrence of replicated variants within genes such as FLT3 and KMT2A. --enable-variant-deduplication true should be used to filter all small indels in the structural variant VCF that appear and are passing in the small variant VCF, PASS in the FILTER column of the small variant VCF file. DRAGEN normalizes variants by trimming and left shifting up to 500 bases. The new deduplicated SV VCF file will have the same prefix passed by --output-file-prefix followed by sv.small_indel_dedup.

Somatic WGS Tumor Only

This recipe is for processing whole genome sequencing data for somatic tumor only workflows.

Example Command Line

For most scenarios, simply creating the union of the command line options from the single caller scenarios will work.

- Configure the INPUT options
- Configure the OUTPUT options
- Configure MAP/ALIGN depending on if realignment is desired or not
- Configure the VARIANT CALLERs based on the application
- Configure any additional options

- Build up the necessary options for each component separately, so that they can be re-used in the final command line.

The following are partial templates that can be used as starting points. Adjust them accordingly for your specific use case:

```
#!/bin/bash

set -euo pipefail


# Path to DRAGEN hashtable
DRAGEN_HASH_TABLE=<REF_DIR>

# Path to output directory for the DRAGEN run
OUTPUT=<OUT_DIR>

# File prefix for DRAGEN output files
PREFIX=<OUT_PREFIX>

# Population SNP VCF. It can be retrieved from catalogs of population variation
# such as the 1000 genome project or other large cohort discovery efforts.
# Only high-frequency SNPs should be included. A suitable file can be retrieved
# from the GATK resource bundle: 1000G_phase1.snps.high_confidence.vcf.gz
CNV_POP_VCF=<POPULATION_VCF_PATH>

# Path to VC systematic noise BED file. In tumor-only variant calling, this filter
# is essential for removing systematic noise observed in normal samples. Prebuilt
# systematic noise files are available for download on the Illumina DRAGEN Bio-IT
# Platform support site page. Alternatively, running the somatic TO pipeline on
```

```
# normal samples can generate a systematic noise file. We recommend using a  
  
# systematic noise file based on normal samples that match the library prep of  
  
# the tumor samples. A prebuilt systematic noise BED file can be downloaded from  
  
# https://support.illumina.com/sequencing/sequencing\_software/dragen-bio-it-platform/product\_files.html  
  
VC_SYSTEMATIC_NOISE_FILE=<VC_SYSTEMATIC_NOISE_BED_FILE_PATH>  
  
  
# The Nirvana annotation database is downloadable at  
  
# https://support-docs.illumina.com/SW/DRAGEN\_v310/Content/SW/DRAGEN/IAE\_DownloadData.htm  
  
NIRVANA_ANNOTATION_FOLDER=<NIRVANA_ANNOTATION_FOLDER_PATH>  
  
  
# Define the input sources, select fastq, bam, or cram.  
  
INPUT_FASTQ_LIST="  
  
    --tumor-fastq-list $TUMOR_FASTQ_LIST \  
  
    --tumor-fastq-list-sample-id $TUMOR_FASTQ_LIST_SAMPLE_ID \  
  
"  
  
  
INPUT_FASTQ="  
  
    --tumor-fastq1 $TUMOR_FASTQ1 \  
  
    --tumor-fastq2 $TUMOR_FASTQ2 \  
  
    --RGSM-tumor $RGSM_TUMOR \  
  
    --RGID-tumor $RGID_TUMOR \  
  
"  
  
  
INPUT_BAM="
```

```
--tumor-bam-input $TUMOR_BAM \
"
INPUT_CRAM=""
--tumor-cram-input $TUMOR_CRAM \
"
# Select input source, here in this example we use INPUT_FASTQ_LIST
INPUT_OPTIONS=""
--ref-dir $DRAGEN_HASH_TABLE \
$INPUT_FASTQ_LIST \
"
OUTPUT_OPTIONS=""
--output-directory $OUTPUT \
--output-file-prefix $PREFIX \
"
MA_OPTIONS=""
--enable-map-align true \
--enable-sort true \
--enable-duplicate-marking true \
"
CNV_OPTIONS=""
```

```
--enable-cnv true \
--cnv-population-b-allele-vcf $CNV_POP_VCF \
"
SNV_OPTIONS="

--enable-variant-caller true \
--vc-systematic-noise $VC_SYSTEMATIC_NOISE_FILE \
--vc-enable-germline-tagging true \
--enable-variant-annotation true \
--variant-annotation-data $NIRVANA_ANNOTATION_FOLDER \
--variant-annotation-assembly $REFERENCE \
"
SV_OPTIONS="

--enable-sv true \
--sv-systematic-noise $SV_SYSTEMATIC_NOISE_BEDPE \
"
SNV_SV_DEDUPPLICATION_OPTIONS="

--enable-variant-deduplication true \
"
#
# Construct final command line
CMD="

dragen \

```

```
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \
$MA_OPTIONS \
$CNV_OPTIONS \
$SNV_OPTIONS \
$SV_OPTIONS \
$SNV_SV_DEDUPLICATION_OPTIONS \
"
# Execute
echo $CMD
bash -c $CMD
```

Optional Settings

Optional settings per component are listed below. Refer to [Command-line Options on page 54](#) for a full list of options.

In general (for most libraries and sample types) we recommend the default values, however for some specific libraries or sample types where it may be advisable to use different values those are explicitly listed below each variant caller section under "library specific settings".

CNV

Option	Description
--cnv-filter-length \$CNV_FILTER_LENGTH	Specifies the minimum event length in bases at which a reported event is marked as PASS in the output VCF file.
--cnv-merge-distance \$CNV_MERGE_DISTANCE	Specifies the maximum segment gap allowed for merging segments.

Option	Description
--cnv-somatic-enable-het-calling true	Enable HET-calling mode for heterogeneous segments. For more information, see HET-calling Mode on page 223 .

CNV Library Specific Settings

Include the matched normal sample in the CNV panel of normals

Option	Liquid Tumors (e.g., AML/MLL)
--cnv-filter-length \$CNV_FILTER_LENGTH	500000
--cnv-merge-distance \$CNV_MERGE_DISTANCE	20000000
--cnv-somatic-enable-het-calling true	True

SNV

Option	Description
--vc-sq-filter-threshold \$THRESHOLD	Threshold for sensitivity-specificity tradeoff. The default threshold is 4(Solid)/2(Liquid). Raise this value to improve specificity at the cost of sensitivity, or lower it to improve sensitivity at the cost of specificity.

Option	Description
--vc-systematic-noise \$SYSTEMATIC_NOISE_FILE	Systematic noise filter. This filter can be useful to remove systematic noise observed in normal samples. Prebuilt systematic noise files are available for download on the Illumina DRAGEN Bio-IT Platform support site page. Alternatively, a systematic noise file can be generated by running the somatic TO pipeline on normal samples. We recommend using a systematic noise file based on normal samples that match the library prep of the tumor samples.
--vc-somatic-hotspots somatic_hotspots_GRCh38.vcf.gz	Hotspots file. By default, DRAGEN treats positions in the COSMIC database as hotspots, assigning an increased prior probability to variants at these positions. Use this option to override with a custom hotspots file if a list of positions of interest is available.
--vc-combine-phased-variants-distance \$DIST	Combining phased variants. By default, DRAGEN will not combine nearby phased calls into MNVs or indels. To combine such calls, set this parameter to a value greater than zero indicating the maximum distance at which calls should be combined. If the user wants to enable the combining of phased variants the recommended value of the distance is 15 base pairs. The valid range is [0; 15]

Option	Description
<pre>--vc-enable-germline-tagging true --enable-variant-annotation true --variant-annotation-data \$NIRVANA_ANNOTATION_FOLDER --variant-annotation-assembly \$REFERENCE</pre>	Germline filtering. Enable to tag variants as germline or somatic based on population databases. \$REFERENCE can be GRCh37 or GRCh38 (GRCh37 is compatible with hs37d5 and hg19). The Nirvana annotation database is downloadable at this page.
<pre>--vc-target-vaf FLOAT</pre>	This option is available with DRAGEN v4.2+. The vc-target-vaf is used to select the variant allele frequencies of interest. The variant caller will aim to detect variants with allele frequencies equal to and larger than this setting. This setting will not apply a hard filter and it is possible to detect variants with allele frequencies lower than the selected threshold. On high coverage and clean datasets, a lower target-vaf may help increase sensitivity. On noisy samples (like FFPE) a higher target-vaf maybe help reduce false positives. Using a low target-vaf may also increase runtime. The valid range is [0, 1]. The default is 0.03 (or 0.001 when --vc-enable-umi-liquid=true).

SNV library specific settings

Option	Description
--vc-target-vaf FLOAT	In FFPE samples with UMI Simplex collapsing it may be beneficial to increase the vc-target-vaf to 0.2 or 0.3. In FFPE samples with UMI Duplex collapsing some of the strand specific FFPE deamination noise may be removed by the duplex collapsing so that the default vc-target-vaf of 0.01 may remain appropriate.
--vc-excluded-regions-bed \$BED	Some FFPE samples may have a high rate of FP calls in SINE (and specifically in ALU) regions. Optionally use an ALU bed to hard filter all calls in this region. Steps are provided below to download an ALU region bed.

Build the SNV systematic noise file

- Run DRAGEN somatic tumor-only on each of approximately 50 normal samples:

```
### choose input either from
### i) BAM
INPUT="--tumor-bam-input ${NORMAL_BAM}"
### ii) FASTQs
INPUT="--tumor-fastq-list ${NORMAL_FASTQ_LIST} \
--tumor-fastq-list-sample-id ${NORMAL_FASTQ_LIST_SAMPLE_ID}"
###

dragen \
```

```

-r ${REFERENCE} \
${INPUT} \
--enable-variant-caller true \
--vc-detect-systematic-noise true \
--build-sys-noise-germline-vaf-threshold=1 \
--vc-enable-germline-tagging true \
--enable-variant-annotation true \
--variant-annotation-data ${NIRVANA_ANNOTATION_FOLDER} \
--variant-annotation-assembly ${REFERENCE} \
--intermediate-results-dir ${TMP} \
--output-directory ${DIR} \
--output-file-prefix ${PREFIX}

```

2. Gather the full paths to the VCF files from the previous step, and create \${VCF_LIST} by specifying 1 file path per line.
3. Generate the final noise file

```

dragen \
-r ${REFERENCE} \
--build-sys-noise-vcfs-list ${VCF_LIST} \
--build-sys-noise-method max \
--intermediate-results-dir ${TMP} \
--output-directory ${DIR} \
--output-file-prefix ${PREFIX}

```

Download a SINE/ALU regions bed for SNV excluded regions

ALUs comprise approximately 11% of the genome and are common in introns. High rates of deamination FP calls have been observed in some FFPE libraries. If the ALU regions are not clinically significant for a specific analysis, use the following steps to filter out the entire ALU region.

1. visit genome.ucsc.edu
2. select the appropriate reference: hg19/hg38
3. specify the following options:
 - Output format: BED
 - Output file name: \${PREFIX}.bed
4. Download the file
5. Open the file and remove redundant headers
6. The 4th column specifies the SINE family. Grep the file using Alu to limit the file to only ALU region
7. Set the excluded regions filter: --vc-excluded-regions-bed \$BED

SV

See [Liquid Tumor Calling on page 302](#) for more information.

Option	Description
--sv-enable-liquid-tumor-mode true	Enable liquid tumor mode.
--sv-tin-contam-tolerance \$TIN_CONTAM_TOLERANCE	Set the Tumor-in-Normal (TiN) contamination tolerance level.

SV library specific settings

Option	Description
--sv-systematic-noise \$SV_SYSTEMATIC_NOISE_BEDPE	A prebuilt systematic noise BEDPE file can be downloaded from Product Files .
--sv-min-scored-variant-size \$MIN_SCORED_VAR_SIZE	100000

cfDNA UMI with Enrichment on FFPE Samples

This recipe is for processing sequencing data with Unique Molecular Identifier (UMI) for Illumina cell-free DNA Prep with Enrichment kit for formalin-fixed, paraffin-embedded formalin-fixed, paraffin-embedded (FFPE) samples.

Example Command Line

For most scenarios, creating the union of the command line options from the single caller scenarios works.

- Configure the INPUT options
- Configure the OUTPUT options
- Configure the VARIANT CALLERs based on the application
- Configure any additional options
- Build up the necessary options for each component separately, so that they can be re-used in the final command line

The following are partial templates that can be used as starting points. Adjust them accordingly for your specific use case.

```
#!/bin/bash

set -euo pipefail


# Path to DRAGEN hashtable
DRAGEN_HASH_TABLE=<REF_DIR>

# Path to output directory for the DRAGEN run
OUTPUT=<OUT_DIR>

# File prefix for DRAGEN output files
PREFIX=<OUT_PREFIX>

# Target BED
TARGET_BED=<MANIFEST.bed>
```

```
# Define the input sources, select fastq list, fastq.

INPUT_FASTQ_LIST="

--tumor-fastq-list $TUMOR_FASTQ_LIST \
--tumor-fastq-list-sample-id $TUMOR_FASTQ_LIST_SAMPLE_ID \
"

# Select input source, here in this example we use INPUT_FASTQ_LIST

INPUT_OPTIONS="

--ref-dir $DRAGEN_HASH_TABLE \
$INPUT_FASTQ_LIST \
"

OUTPUT_OPTIONS="

--output-directory $OUTPUT \
--output-file-prefix $PREFIX \
--output-format BAM \
"

UMI_OPTIONS="

--enrichment-high-sensitivity-umi true \
--umi-metrics-interval $TARGET_BED \
--qc-coverage-ignore-overlaps true \
--umi-min-supporting-reads 2 \    // Set to 1 for lower sequencing depth
--allow-overrides true \
"
```

```
"  
  
SNV_OPTIONS="  
  
    --vc-target-bed $TARGET_BED \  
    --enable-variant-caller true \  
    --vc-use-somatic-hotspots true \  
    --enrichment-high-sensitivity-snv true \  
    --vc-sq-filter-threshold 0.7 \  
    --vc-enable-umi-solid true          //solid for FFPE samples  
  
"  
  
  
  
CNV_OPTIONS="  
  
    --enrichment-high-sensitivity-cnv true \  
    --cnv-combined-counts <ALL_SAMPLES.COMBINED.COUNTS.txt.gz> \  
    --cnv-target-bed $TARGET_BED \  
  
"  
  
  
  
SV_OPTIONS="  
  
    --enrichment-high-sensitivity-sv false \  
    --sv-tumor-only-min-pass-somatic-score 0 \  
    --sv-locus-node-target-file $TARGET_BED \  
  
"  
  
  
  
ANNOTATION_SETTINGS="  
  
"
```

```

--enable-variant-annotation false \\           // Default to false
--vc-enable-germline-tagging true \\           // Only if annotation is enabled
--vc-log-insert-size true \\           // Only if annotation is enabled
--germline-tagging-db-threshold 10 \\           // Only if annotation is enabled
--tmb-enable-proxi-filter true \\           // Only if annotation is enabled
"
"

QC_SETTINGS="

--qc-coverage-region-1 $TARGET_BED \
--qc-coverage-filters-1 'mapq<0' \
--qc-coverage-region-2 $TARGET_BED \
--qc-coverage-filters-2 'mapq<0' \
--qc-coverage-region-padding-2 150 \
--qc-coverage-reports-1 cov_report full_res \
--enable-metrics-json true \
--watchdog-verbose-logging true \
--force \
"
"

# Construct final command line

CMD="

Dragen \
$INPUT_OPTIONS \
$OUTPUT_OPTIONS \

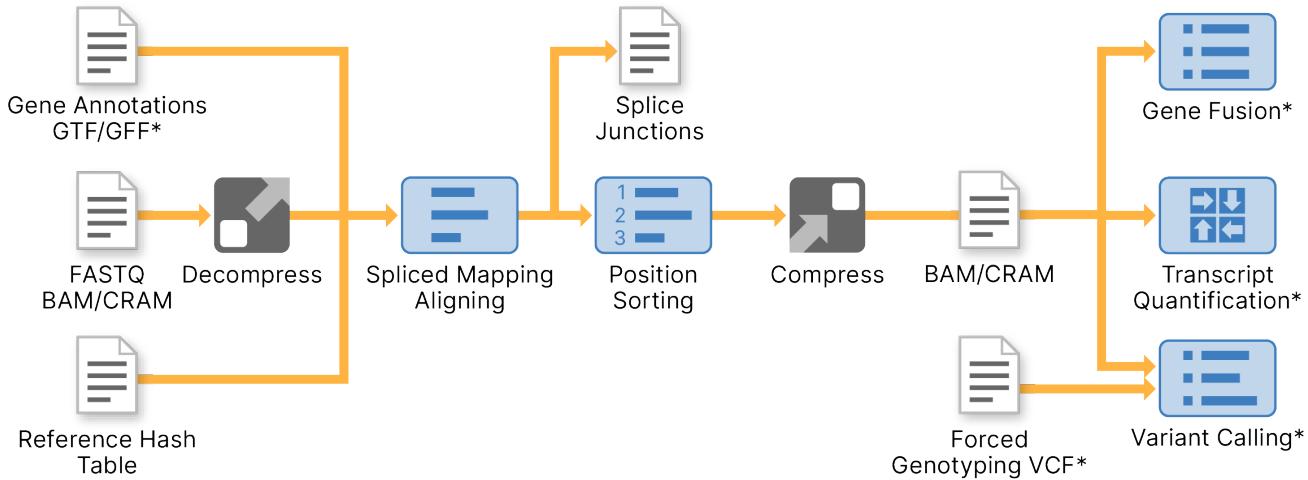
```

```
$SNV_OPTIONS \
$CNV_OPTIONS \
$SV_OPTIONS \
$ANNOTATION_SETTINGS \
$QC_SETTINGS \
"

# Execute
Echo $CMD
Bash -c $CMD
```

DRAGEN RNA Pipeline

DRAGEN includes an RNA-seq (splicing-aware) aligner, as well as RNA specific analysis components for gene expression quantification and gene fusion detection.



* Optional

Most of the functionality and options described in Host Software Options and DNA Mapping also apply to RNA applications. Additional RNA-specific aspects are described in this section.

Input Files

Gene Annotation File

In addition to the standard input files (reads from FASTQ or BAM, reference genome, etc.), DRAGEN can also take a gene annotations file as input. A gene annotations file aids in the alignment of reads to known splice junctions, and is required for gene expression quantification and gene fusion calling.

To specify a gene annotation file, use the `-a (--annotation-file)` command line option. The input file must conform to the GTF/GFF specification. See [GFF/GTF File Format](#) for more information. The file must contain features of type exon, and the record must contain attributes of type `gene_id` and `transcript_id`. An example of a valid GTF file is shown below.

```

chr1  HAVANA transcript 11869 14409 . + . gene_id "ENSG0000223972.4";
transcript_id ENST0000456328.2; ...

chr1  HAVANA exon      11869 12227 . + . gene_id "ENSG0000223972.4";
transcript_id ENST0000456328.2; ...
  
```

```

chr1  HAVANA exon      12613  12721  .  +  .  gene_id "ENSG00000223972.4";
transcript_id ENST00000456328.2; ...

chr1  HAVANA exon      13221  14409  .  +  .  gene_id "ENSG00000223972.4";
transcript_id ENST00000456328.2; ...

chr1  ENSEMBL transcript 11872  14412  .  +  .  gene_id "ENSG00000223972.4";
transcript_id ENST00000515242.2; ...

chr1  ENSEMBL exon      11872  12227  .  +  .  gene_id "ENSG00000223972.4";
transcript_id ENST00000515242.2; ...

chr1  ENSEMBL exon      12613  12721  .  +  .  gene_id "ENSG00000223972.4";
transcript_id ENST00000515242.2; ...

chr1  ENSEMBL exon      13225  14412  .  +  .  gene_id "ENSG00000223972.4";
transcript_id ENST00000515242.2; ...

...

```

Similarly, a GFF file can be used. Each exon feature must have as a Parent a transcript identifier that is used to group exons. An example of a valid GFF file is shown below.

```

1  ensembl_havana processed_
transcript 11869  14409  .  +  .  ID=transcript:ENST00000456328;

1  havana      exon      11869  12227  .  +  .  Parent=transcri
pt:ENST00000456328; ...

1  havana      exon      12613  12721  .  +  .  Parent=transcri
pt:ENST00000456328; ...

1  havana      exon      13221  14409  .  +  .  Parent=transcri
pt:ENST00000456328; ...

...

```

In order to distinguish exons within the PAR region of chromosome X from within the PAR region of chromosome Y, the `gene_id` attribute of all exons of the same gene must be distinct between the two chromosomes. Often case, the `gene_id` of all exons of a transcript from geneA is equal to `gene_id=geneA` in chromosome X, and `gene_id=geneA_PAR_Y` in chromosome Y. It allows the GTF/GFF parser and downstream components to discriminate data associated with PAR genes in chromosome X from data associated with the same PAR genes in chromosome Y.

The DRAGEN host software parses the file for exons within the transcripts and produces splice junctions. The following output displays the number of splice junctions detected.

```
=====
Generating annotated splice junctions
=====

Input annotations file: ./gencode.v19.annotation.gtf
Splice junctions database file: output/rna.sjdb.annotations.out.tab

Number of genes: 27459

Number of transcripts: 196520

Number of exons: 1196293

Number of splice junctions: 343856
```

The splice junctions that are detected from the annotation file are also written to `*.sjdb.annotations.out.tab`. Splice junctions below a minimum length are excluded, which helps filter annotation artifacts that do not meet the minimum required length. This helps to lower the false detection rate for falsely annotated junctions. This minimum annotation splice junction length is controlled by the `--rna-ann-sj-min-len` option, which has a default value of 6.

Two-Pass Splice Junction Alignment

Instead of using a GTF file for annotated splice junctions, the DRAGEN software is also capable of reading in an `SJ.out.tab` file (see [SJ.out.tab on page 1](#)). This file enables DRAGEN to run in a two-pass mode, where the splice junctions discovered in the first pass (output as `SJ.out.tab` file) are used to guide the mapping and alignment reads during a second run through DRAGEN. This mode of operation is useful to increase sensitivity for spliced alignments in cases when a gene annotations file is not readily available for the target genome. If a well curated GTF is already available for your target genome, then there is no need to run a second pass with the `SJ.out.tab`.

Depending on the characteristics of the input file (for example, read depth and distribution) the second pass using the first pass `SJ.out.tab` may take longer than the first pass.

-  Components downstream of aligner like gene expression quantification, gene fusion detection and RNA variant calling require GTF file as the input annotations file and are NOT compatible with two-pass splice-junction alignment mode.

RNA Alignment

The DRAGEN RNA pipeline uses the DRAGEN RNA spliced aligner. Mapping of short seed sequences from RNA reads is performed similarly to mapping DNA reads. In addition, splice junctions (the joining of noncontiguous exons in RNA transcripts) near the mapped seeds are detected and incorporated into the full read alignments.

Alignment Output

When using the RNA workflow, the output files generated are similar to the files generated by the DNA workflow. The RNA workflow also produces extra information related to spliced alignments. Details regarding the splice junctions are present both in the SAM alignment record and an additional file, the SJ.out.tab file.

BAM Tags

The output BAM file meets the SAM specification and is compatible with downstream RNA analysis tools. The following BAM tags are emitted alongside spliced alignments.

- XS:A—The XS tag denotes the strand orientation of an intron. Refer to [Compatibility with Cufflinks on page 541](#).
- jM:B—The jM tag lists the intron motifs for all junctions in the alignments. It has the following definitions:
 - 0: non-canonical
 - 1: GT/AG
 - 2: CT/AC
 - 3: GC/AG
 - 4: CT/GC
 - 5: AT/AC
 - 6: GT/AT

If a gene annotations file is used during the map/align stage, and the splice junction is detected as an annotated junction, then 20 is added to its motif value.

- NH:i—A standard SAM tag indicating the number of reported alignments that contains the query in the current record. This tag may be used for downstream tools such as featureCounts.
- HI:i—A standard SAM tag denoting the query hit index, with its value indicating that this alignment is the i -th one stored in the SAM. Its value ranges from 1 ... NH. This tag may be used for downstream tools such as featureCounts.

Compatibility with Cufflinks

Cufflinks might require spliced alignments to emit the XS:A strand tag. This tag is present in the SAM record if the alignment contains a splice junction. The possible values for XS:A strand tag are as follows:

‘.’ (undefined), ‘+’ (forward strand), ‘-’ (reverse strand), or ‘*’ (ambiguous).

By default, if the spliced alignment has an undefined strand or an ambiguous (conflicting) strand, then the alignment output is suppressed.

The alignments can be output into the output alignment file by setting the `--no-ambig-strand` option to 1. Cufflinks also expects that the MAPQ for a uniquely mapped read is a single value. This value is specified by the `--rna-mapq-unique` option. To force all uniquely mapped reads to have a MAPQ equal to this value, set `--rna-mapq-unique` to a nonzero value.

SJ.out.tab

Along with the alignments emitted in the SAM/BAM file, an additional SJ.out.tab file summarizes the high confidence splice junctions in a tab-delimited file. The columns for this file are as follows:

1. contig name
2. first base of the splice junction (1-based)
3. last base of the splice junction (1-based)strand (0: undefined, 1: +, 2: -)
4. strand (0: undefined, 1: +, 2: -)
5. intron motif: 0: noncanonical, 1: GT/AG, 2: CT/AC, 3: GC/AG, 4: CT/GC, 5: AT/AC, 6: GT/AT
6. 0: unannotated, 1: annotated, only if an input gene annotations file was used
7. number of uniquely mapping reads spanning the splice junction
8. number of multimapping reads spanning the splice junction
9. maximum spliced alignment overhang

The maximum spliced alignment overhang (column 9) field in the SJ.out.tab file is the anchoring alignment overhang. For example, if a read is spliced as ACGTACGT-----ACGT, then the overhang is 4. For the same splice junction, across all reads that span this junction, the maximum overhang is reported. The maximum overhang is a confidence indicator that the splice junction is correct based on anchoring alignments.

There are two SJ.out.tab files generated by the DRAGEN host software, an unfiltered version and a filtered version. The records in the unfiltered file are a consolidation of all spliced alignment records from the output SAM/BAM. However, the filtered version has a much higher confidence for being correct due to the use of the following filters.

A splice junction entry in the SJ.out.tab file is filtered out if *any* of these conditions are met:

- SJ is a noncanonical motif and is only supported by < 3 unique mappings.
- SJ of length > 50000 and is only supported by < 2 unique mappings.
- SJ of length > 100000 and is only supported by < 3 unique mappings.

- SJ of length > 200000 and is only supported by < 4 unique mappings.
- SJ is a noncanonical motif and the maximum spliced alignment overhang is < 30.
- SJ is a canonical motif and the maximum spliced alignment overhang is < 12.

The filtered SJ.out.tab is recommended for use with any downstream analysis or post processing tools. Alternatively, you can use the unfiltered SJ.out.tab and apply your own filters (for example, with basic awk commands).

Note that the filter does not apply to the alignments present in the BAM or SAM file.

Mapping Metrics

The RNA Pipeline reports summary and per read group statistics pertaining to read mapping in the mapping_metrics.csv file. The majority of the metrics are as described in the [Mapping and Aligning Metrics on page 328](#) section, and the RNA-Seq specific metrics are listed below.

Filter	Description
Filtered rRNA reads	Total number of ribosomal RNA reads that are filtered out with the --rrna-filter-enable option.
Mitochondrial reads excluded	Total number of reads detected to be in ChrM if the --rna-mapping-metrics-exclude-chrm option is enabled.
Mapped reads adjusted for filtered mapping	Adjusted count of mapped reads by adding in the filtered rRNA reads.
Mapped reads adjusted for excluded mapping	Adjusted count of mapped reads by adding in the excluded mitochondrial reads.
Mapped reads adjusted for filtered and excluded mapping	Adjusted count of mapped reads by adding in both the filtered rRNA and excluded mitochondrial reads.
Unmapped reads adjusted for filtered mapping	Adjusted count of unmapped reads by subtracting out the filtered rRNA reads.

Filter	Description
Unmapped reads adjusted for excluded mapping	Adjusted count of unmapped reads by subtracting out the excluded mitochondrial reads.
Unmapped reads adjusted for filtered and excluded mapping	Adjusted count of unmapped reads by subtracting out both the filtered rRNA and the excluded mitochondrial reads.
Reads with splice junction	Total number of reads that included a spliced alignment that crosses an intron.

Chimeric.out.junction File

If there are chimeric alignments present in the sample, then a supplementary Chimeric.out.junction file is also output. This file contains information about split-reads that can be used to perform downstream gene fusion detection. Each line contains one chimerically aligned read. The columns of the file are as follows:

1. Chromosome of the donor.
2. First base of the intron of the donor (1-based).
3. Strand of the donor.
4. Chromosome of the acceptor.
5. First base of the intron of the acceptor (1-based).
6. Strand of the acceptor.
7. N/A—not used, but is present to be compatible with other tools. It will always be 1.
8. N/A—not used, but is present to be compatible with other tools. It will always be *.
9. N/A—not used, but is present to be compatible with other tools. It will always be *.
10. Read name.
11. First base of the first segment, on the + strand.
12. CIGAR of the first segment.
13. First base of the second segment.
14. CIGAR of the second segment.

CIGARs in this file follow the standard CIGAR operations as found in the SAM specification, with the addition of a gap length L that is encoded with the operation p. For paired end reads, the sequence of the second mate is always reverse complemented before determining strandedness.

The following is an example entry that shows two chimerically aligned read pairs, in which one of the mates is split, mapping segments of chr19 to chr12. Also shown are the corresponding SAM records associated with these entries.

```

chr19 580462 + chr12 120876182 + 1 * * R_15448 571532
49M8799N26M8p49M26S 120876183 49H26M
chr19 580462 + chr12 120876182 + 1 * * R_15459 571552
29M8799N46M8p29M46S 120876183 29H46M

R_15448:1    99      chr19    571531      60      49M8799N26M   =
580413
R_15448:2    147     chr19    580413      60      49M26S       =
571531
R_15448:2    2193    chr12    120876182    15      49H26M       chr19
571531

R_15459:1    99      chr19    571551      60      29M8799N46M   =
580433
R_15459:2    147     chr19    580433      4       29M46S       =
571551
R_15459:2    2193    chr12    120876182    15      29H46M       chr19
571551

```

RNA Alignment Options

The aligner stage of the RNA spliced aligner uses Smith-Waterman Alignment Scoring options and Splicing Scoring Options.

Smith-Waterman Alignment Scoring Options

Refer to [Smith-Waterman Alignment Scoring Settings on page 1](#) for more details about the alignment algorithm used within DRAGEN. The following scoring options are specific to the processing of canonical and noncanonical motifs within introns.

Option	Description
--Aligner.intron-motif12-pen	The --Aligner.intron-motif12-pen option controls the penalty for canonical motifs 1/2 (GT/AG, CT/AC). The default value calculated by the host software is 1*(match-score + mismatch-pen).
--Aligner.intron-motif34-pen	The --Aligner.intron-motif34-pen option controls the penalty for canonical motifs 3/4 (GC/AG, CT/GC). The default value calculated by the host software is 3*(match-score + mismatch-pen).
--Aligner.intron-motif56-pen	The --Aligner.intron-motif56-pen option controls the penalty for canonical motifs 5/6 (AT/AC, GT/AT). The default value calculated by the host software is 4*(match-score + mismatch-pen).
--Aligner.intron-motif0-pen	The --Aligner.intron-motif0-pen option controls the penalty for noncanonical motifs. The default value calculated by the host software is 6*(match-score + mismatch-pen).

Splicing Scoring Options

Option	Description
--Mapper.min-intron-bases	For RNA-Seq mapping, a reference alignment gap can be interpreted as a deletion or an intron. In the absence of an annotated splice junction, the min-intron-bases option is a threshold gap length separating this distinction. Reference gaps at least this long are interpreted and scored as introns, and shorter reference gaps are interpreted and scored as deletions. However, alignments can be returned with annotated splice junctions shorter than this threshold.
--Mapper.max-intron-bases	The max-intron-bases option controls the largest possible intron that is reported, which useful for preventing false splice junctions that would otherwise be reported. Set this option to a value that is suitable to the species you are mapping against.
--Mapper.ann-sj-max-indel	For RNA-seq, seed mapping can discover a reference gap in the position of an annotated intron, but with slightly different length. If the length difference does not exceed this option, the mapper investigates the possibility that the intron is present exactly as annotated, but an indel on one side or the other near the splice junction explains the length difference. Indels longer than this option and very near annotated splice junctions are not likely to be detected. Higher values may increase mapping time and false detections.

Duplicate Marking

DRAGEN RNA can detect duplicate reads, which are defined as fragments that have both ends mapping to the same (clipping-adjusted) position during alignment. In RNA data, the reads can represent PCR duplicates during library prep or as a result from deep coverage of highly expressed regions. If `--enable-duplicate-marking` is set to true, duplicate fragments are marked in the BAM file and the total number of duplicate reads is reported as a mapping metric. Marking of duplicates does not affect gene expression quantification and gene fusion calling.

Downsampling

DRAGEN RNA also supports internal downsampling, which is a process by which a random sub-sample of reads is selected from the dataset after trimming and alignment for downstream analysis. In RNA-Seq, this can be useful in two ways - it can speed up the analysis of samples with excessively high coverage, and it can allow for more accurate cross-comparisons between different samples.

If `--enable-down-sampler` is set to true and a value specified for `--down-sampler-reads`, DRAGEN will use only that many RNA fragments (including both Read 1 and Read 2) for quantification, fusion, variant calling and output to BAM. The entire input dataset is still used for the generation of trimming, fastqc, and mapping metrics.

Ribosomal RNA Filtering

Ribosomal RNA (rRNA) sequences can contribute a large fraction of reads in some RNA datasets, depending on the sample type and library prep method. You can use the DRAGEN RNA pipeline to filter rRNA reads during alignment, because the reads are not relevant for downstream analysis. By filtering rRNA, you can reduce run time and file size and avoid deep read alignment pile ups at rRNA repeat loci on the genome to make downstream analysis of RNA BAM files easier.

rRNA filtering relies on a decoy contig with the rRNA sequence included in the reference hash table. Any read that maps to the decoy contig, including multimappers, is tagged with rRNA and is not mapped in the output.

Command-Line Options

The following are the required command-line options for rRNA filtering.

- `--rrna-filter-enable=true`—Enables rRNA filtering. Set to `true` to enable rRNA filtering. The default value is `false`.
- `--rrna-filter-contig`—Specify the name of the rRNA sequences to use for filtering. If you don't specify a value, the default `g1000220` is provided for human genome alignments by using the reference autodetect feature. `g1000220` is an unplaced contig included in hg19 and hg38 genomes, which include a full copy of the rRNA repeat. For other genomes, you must include a rRNA decoy contig when creating a hash table.

Output Files

All rRNA filtering reads are left unaligned in the BAM files and tagged `ZS:Z:FLT`. The number and percentage of filtered rRNA reads is reported as a mapper metric `Adjustment of reads matching filter contigs`.

PolyA Trimming

PolyA tails may be trimmed by including the settings `--read-trimmers polya` or `--soft-read-trimmers polyg,polya`. Polyg soft trimming is enabled by default. The minimum number of poly-A/poly-T bases required for trimming may be set using `--trim-polya-min-trim`. The default threshold is 20 poly-A/poly-T bases. Refer to [Read Trimming on page 92](#) for usage of read trimmers options.

PolyA trimming by read orientation

The PolyA trimmer determines which end of the reads to trim for poly-A and poly-T sequences based on the library type. For example for Illumina forward stranded paired reads the trimmer will trim poly-A sequences at 3' end of read 1 and poly-T sequences at 5' end of read 2. If `--rna-library-type` is not provided or set to auto detect (A), the trimmer assumes the library is not stranded and trims poly-A sequences from 3' end of each read and poly-T sequences from 5' end of each read. The option `--rna-library-type` is described in [Gene Expression Quantification](#).

MAPQ Scoring

By default, the MAPQ calculation for RNA-Seq is identical to DNA-Seq. The primary contributor to MAPQ calculation is the difference between the best and second-best alignment scores. Therefore, adjusting the alignment scoring parameters impacts the MAPQ estimate. These adjustments are outlined in [Smith-Waterman Alignment Scoring Settings on page 1](#).

The `--mapq-strict-sjs` option is specific to RNA, and applies where at least one exon segment is aligned confidently, but there is ambiguity regarding possible splice junctions. When this option is set to 0, a higher MAPQ value is returned, expressing confidence that the alignment is at least partially correct. When this option is set to 1, a lower MAPQ value is returned, reflecting the splice junction ambiguity.

Some downstream tools, such as Cufflinks, expect the MAPQ value to be a unique value for all uniquely mapped reads. This value is specified with the `--rna-mapq-unique` option. Setting this option to a nonzero value overrides all MAPQ estimates based on alignment score. Instead, all uniquely mapped reads have a MAPQ set to the value of `--rna-mapq-unique`. All multimapped reads have a MAPQ value of $\text{int}(-10 * \log_{10}(1 - 1/\text{NH}))$, where the NH value is the number of hits (primary and secondary alignments) for that read.

Gene Fusion Detection

The DRAGEN Gene Fusion module uses the DRAGEN RNA spliced aligner for detection of gene fusion events. It performs a split-read analysis on the supplementary (chimeric) alignments to detect potential breakpoints. The putative fusion events then go through various filtering stages to mitigate potential false positives. In addition to the final results, all potential candidates (unfiltered) are output, which can be used to maximize sensitivity.

Running DRAGEN Gene Fusion

You can run the DRAGEN Gene Fusion module together with a regular RNA-Seq map/align job. To enable the DRAGEN Gene Fusion module, set `--enable-rna-gene-fusion` to true in your current RNA-Seq command-line scripts. The DRAGEN Gene Fusion module requires a gene annotations file in GTF or GFF format.

The following is an example command line for running an end to end RNA-Seq experiment.

```
/opt/edico/bin/dragen \
-r <HASHTABLE> \
-1 <FASTQ1> \
-2 <FASTQ2> \
-a <GTF_FILE> \
--output-dir <OUT_DIRECTORY> \
--output-file-prefix <PREFIX> \
--RGID <READ_GROUP_ID> \
--RGSM <Sample_NAME> \
--enable-rna true \
--enable-rna-gene-fusion true
```

At the end of a run, a summary of detected gene fusion events is output, which is similar to the following example.

```
=====
Loading gene annotations file
=====

Input annotations file: ref_annot.gtf
Number of genes: 27459
Number of transcripts: 196520
Number of exons: 1196293
=====

Launching DRAGEN Gene Fusion Detection
=====

Min nonintact split support 3
```

```

rna-gf-blast-pairs: blast_pairs.outfmt6
rna-gf-min-blast-pairs-eval: 1e-100
rna-gf-exon-snap: 50
rna-gf-coverage-lookup-window: 1000
rna-gf-min-support: 2
rna-gf-min-support-be: 10
rna-gf-min-breakpoint-mapq: 20
rna-gf-min-unique-alignments: 2
rna-gf-restrict-genes true
=====
Completed DRAGEN Gene Fusion Detection
=====
Chimeric alignments: 683343
Total fusion candidates: 2370
Final fusion candidates: 26
RUN TIME Time loading reference 00:00:37.696 37.70
RUN TIME Time loading anchor reference 00:00:36.720 36.72
RUN TIME Time loading gene annotations 00:00:04.784 4.78
RUN TIME Time aligning reads 00:00:57.370 57.37
RUN TIME Time aligning anchored reads 00:00:21.931 21.93
RUN TIME Time merging anchored reads 00:00:21.546 21.55
RUN TIME Time duplicate marking 00:00:05.812 5.81
RUN TIME Time sorting and marking duplicates 00:01:00.949 60.95
RUN TIME Time saving map/align output 00:01:31.879 91.88
RUN TIME Time running gene fusion event generation 00:00:00.943 0.94
RUN TIME Time running gene fusion filter 00:00:00.612 0.61
RUN TIME Time partitioning 00:02:00.800 120.80
RUN TIME Total runtime 00:04:38.725 278.73
*****
DRAGEN finished normally

```

Run Gene Fusion Standalone

The DRAGEN Gene Fusion module can be run as a standalone utility, taking the `*.Chimeric.out.junction` file as input and the gene annotations file as a GTF/GFF file. Running the Gene Fusion module standalone is useful for trying out various configuration options at the gene fusion detection stage, without having to map and align the RNA-Seq data multiple times.

To execute the DRAGEN Gene Fusion module as a standalone utility, use the `--rna-gf-input-file` option to specify the already generated `*.Chimeric.out.junction` file.

The following is an example command line for running the gene fusion module as a standalone utility.

```
/opt/edico/bin/dragen \
-a <GTF_FILE> \
--rna-gf-input-file <INPUT_CHIMERIC> \
--output-dir <OUT_DIRECTORY> \
--output-file-prefix <PREFIX> \
--enable-rna true \
--enable-rna-gene-fusion true
```

Standalone mode does not produce identical results to running from reads.

Gene Fusion Output and Filters

The `<outputPrefix>fusion_candidates.features.csv` file lists the detected gene fusion events. The output CSV file includes the following columns. Any additional columns describe additional features of the fusion candidates.

- `#FusionGene`—Parent gene names (in 5' to 3' order of transcript) participating in the fusion. If a fusion breakpoint overlaps multiple genes, the genes with passing genes (or failing genes if no passing candidate exists) are listed by default as separate candidates (rows). To show them as a semi-colon separated gene list on the same row, the option `--rna-gf-merge-calls` can be set to `true` as described in the Options section.
- `Score`—Fusion call confidence score based on the number of supporting split reads and read-pairs as well as other fusion features. The score can be 0 (low confidence) to 1 (high-confidence call).
- `LeftBreakpoint`—Gene 1 breakpoint formatted as `<Chromosome>:<Position>:<Strand>`.
- `RightBreakpoint`—Gene 2 breakpoint formatted as `<Chromosome>:<Position>:<Strand>`.
- `Filter`—Semicolon separated list of filters. Each output is either a Confidence or Information Only filter. The Filter value is `PASS` if none of the confidence filters are triggered. Otherwise, the output value is `FAIL`.

The following are the available filters.

Filter	Type	Description
<code>DOUBLE_BROKEN_EXON</code>	Confidence	If both breakpoints are 50 bp from annotated exon boundaries, then the number of supporting reads do not satisfy a high threshold requirement (≥ 10 supporting reads).

Filter	Type	Description
LOW_MAPQ	Confidence	All fusion supporting read alignments at either of the breakpoints have MAPQ < 20.
LOW_UNIQUE_ALIGNMENTS	Confidence	All fusion supporting read alignments near at least one of the two breakpoints have the same start and end position.
LOW_SCORE	Confidence	The fusion candidate has low probabilistic score (< 0.5) as determined by the features of the candidate.
MIN_SUPPORT	Confidence	The fusion candidate has < 2 fusion supporting read pairs.
UNENRICHED_GENES	Confidence	If an enrichment list is provided, then neither of the two parent genes is enriched. If amplicon mode is enabled, then at least one of the two parents genes is not enriched. See DRAGEN Amplicon Pipeline on page 609 for further information.
READ_THROUGH	Confidence	The breakpoints are cis neighbors (< 200,000 bp) on the reference genome.
MITOCHONDRIAL_GENES	Confidence	The fusion candidate involves mitochondrial genes. Set --rna-gf-filter-chrm false to disable this filter.
ANCHOR_SUPPORT	Information only	Read alignments of fusion supporting reads are (less than 12 bp) at either of the two breakpoints.
HOMOLOGOUS	Information only	The candidate is likely a false candidate generated because the two genes involved have high gene homology.
LOW_ALT_TO_REF	Information only	The number of fusion supporting reads is < 1% of the number of reads supporting the reference transcript at either of the two breakpoints.
LOW_GENE_COVERAGE	Information only	Either of the two breakpoints have less than 125 bp with nonzero read coverage.

A logistic regression model that has been trained on a large set of RNA data is used for each gene fusion event scoring. The remaining columns in this file provide the value of the features that are either used in this logistic regression model, or used for further filtering of the events, to determine a PASS/FAIL result. Each feature is detailed in the following table along with either the associated logistic regression coefficient, or the associated filter. Some of the features also have some notes for clarification.

i | Specific features and column values are subject to change in future DRAGEN versions as more RNA data is analyzed.

Feature	Coefficient	Default Value	Filter Use	Explanatory Notes
SplitScore	LogSplitScore	1.96 2		Combined count of fusion supporting fragments reported as split reads and soft-clipped reads
NumSplitReads	LogNumSplitReads	0.0		Fusion supporting fragments with at least 1 split read alignment. Not used in model because we useSplitScore
NumSoftClippedReads				Fusion supporting fragments with no split read alignment, but at least 1 soft clipped alignment. Included inSplitScoreand includes soft-clipped reads for both Gene1 and Gene2
NumSoftClippedReadsGene1				Fusion supporting fragments with no split read alignment, but at least 1 soft clipped alignment to Gene 1 (informational)
NumSoftClippedReadsGene2				See above (NumSoftClippedReadsGene1) for Gene 2
NumPairedReads	LogNumPairedReads	6.98 9		Fusion supporting fragments such that the 2 reads map fully to Gene1 and Gene2, but no read overlaps the breakpoint
NumRefSplitReadsGene1				Fragments which map fully within Gene 1 such that at least 1 read aligns across the BP (accumulated as Ref reads)

Feature	Coefficient	Default Value	Filter Use	Explanatory Notes
NumRefPairedReadsGene1				Fragments which map fully within Gene 1 such that the 2 reads map fully on the opposite sides of the breakpoint (accumulated as Ref reads)
NumRefSplitReadsGene2				See above (NumRefSplitReadsGene1) for Gene 2
NumRefPairedReadsGene2				See above (NumRefPairedReadsGene1) for Gene 2
AltToRef	LogAltToRef	2.424	LOW_ALT_TO_REF	Ratio of (fusion split + softclipped reads) / max (NumRefSplitReadsGene1,NumRefSplitReadsGene2)
UniqueAlignmentsGene1			LOW_UNIQUE_ALIGNMENTS	Unique (start-end) positions of fusion supporting read alignments to Gene 1 (after dedup)
UniqueAlignmentsGene2			LOW_UNIQUE_ALIGNMENTS	Unique (start-end) positions of fusion supporting read alignments to Gene 2 (after dedup)
MaxMapqGene1			LOW_MAPQ	Maximum MAPQ for reads in Gene 1
MaxMapqGene2			LOW_MAPQ	Maximum MAPQ for reads in Gene 2
CoverageBasesGene1	LogCoverageBases	0.492		Bases in Gene 1 with depth of coverage greater than a threshold (≥ 1) within a certain distance (size 1000bp) of the breakpoint in the direction of the breakpoint strand which is part of the fusion transcript

Feature	Coefficient	Default Value	Filter Use	Explanatory Notes
CoverageBasesGene2	LogCoverageBases	0.492		See above (CoverageBasesGene1) for Gene 2
DeltaExonBoundaryGene1	LogDeltaExonBoundary	1.026		Distance from the Gene 1 breakpoint for the closest fusion supporting alignment (higher distance to boundary lowers score)
DeltaExonBoundaryGene2	LogDeltaExonBoundary	1.026		See above (DeltaExonBoundaryGene1) for Gene 2
IsRestrictedGene1	IsRestricted	9.380		Indicator variable of whether the Gene 1 is tagged as protein coding or lincRNA in the GTF
IsRestrictedGene2	IsRestricted	9.380		Indicator variable of whether the Gene 2 is tagged as protein coding or lincRNA in the GTF
IsEnrichedGene1				If enrichment or amplicon assay, then indicates whether Gene 1 is enriched. If whole transcriptome sequencing, then set to 1 (used in fusion length and coverage calculations)
IsEnrichedGene2				See above (IsEnrichedGene1) for Gene 2
CisDistance		READ_LENGTH		Distance between breakpoints if they are adjacent to each other and on the same strand. Large value (100M) if not a CIS break
BreakpointDistance				Distance between breakpoints if they are adjacent. Large value (100M) if not within same chromosome
GenePairHomologyEval	LogGenePairHomologyEval	0.108		E-value of pairwise BLAST alignment of the parent genes

Feature	Coefficient	Default Value	Filter Use	Explanatory Notes
AnchorLength1	AnchorLength	0.032		Longest alignment of a fusion supporting read to Gene 1
AnchorLength2	AnchorLength	0.032		Longest alignment of a fusion supporting read to Gene 2
FusionLengthGene1				Distance from breakpoint to the end of Gene 1
FusionLengthGene2				Distance from breakpoint to the end of Gene 2
NonFusionLengthGene1				BP distance to the end of transcript not part of the fusion (Informative)
NonFusionLengthGene2				BP distance to the end of transcript not part of the fusion (Informative)
AdditionalGenes1				Additional genes that overlap Gene 1 breakpoint but did not result in a passing fusion call. Column is only reported if fusion candidate merging is enabled.
AdditionalGenes2				Additional genes that overlap Gene 2 breakpoint but did not result in a passing fusion call. Column is only reported if fusion candidate merging is enabled.
Gene1Id				Gene ID reported in the GTF annotation file
Gene2Id				Gene ID reported in the GTF annotation file

Feature	Coefficient	Default Value	Filter Use	Explanatory Notes
Gene1Location				IntactExon: Breakpoint matches exon boundary, BrokenExon: Breakpoint is within an exon but does not match the exon boundary, Intron: Breakpoint is within an intron, Intergenic: Breakpoint does not overlap any gene
Gene2Location				See above (Gene1Location) for Gene 2
Gene1Sense				True if the Gene 1 5' to 3' direction matches the BP order, indicating that the gene is the upstream gene in the fusion transcript (informative)
Gene2Sense				See above (Gene1Sense) for Gene 2
SvEvent				If SV VCF is provided, then semi-colon separated string representation of SV events matching the fusion candidate.
SvType				If SV VCF is provided, then semi-colon separated list of type of each matching SvEvent.
SomaticScore				If SV VCF is provided, then highest SomaticScore value for matching SvEvents.
SvDistance				If SV VCF is provided, then maximum distance between SV breakpoints and fusion breakpoints (if multiple matching SV events, then minimum over all SV Events).

Feature	Coefficient	Default Value	Filter Use	Explanatory Notes
LeftSvDistance				If SV VCF is provided, then distance between left fusion breakpoint and corresponding SV breakpoint (if multiple matching SV events, then minimum over all SV Events).
RightSvDistance				If SV VCF is provided, then distance between right fusion breakpoint and corresponding SV breakpoint (if multiple matching SV events, then minimum over all SV Events).
SvPresent	SvPresent	0.0	0.377	If SV VCF is provided, then 1 if matching SV event is present, else 0.
SvAbsent				If SV VCF is provided, then 1 if no matching SV event is present, else if no SV VCF provided or if matching SV event is provided, then 0.
Intercept		-8.467		Intercept for logistic regression model.

The `<outputPrefix>fusion_candidates.final` file lists each passing fusion along with the read names that support the fusion, including Split Reads, Soft-clipped reads, and Paired (discordant) Reads. These reads can be extracted from the output BAM file and then used to visualize the fusions (ie in IGV).

The `<outputPrefix>fusion_candidates.vcf.gz` file gives the VCF representation for all of the breakpoints for the candidate fusions using structural variant-style BND notation.

The `<outputPrefix>fusion_metrics.csv` provides a simple count of the total number of fusion candidates, those passing the scoring filter, and the number of unique left-right gene combinations that are found.

Gene Fusion Options and Filters

The following options can be used to configure the fusion caller:

Option	Description
--rna-gf-blast-pairs	A file listing gene pairs that have a high level of similarity. This list of gene pairs is used as a homology filter to reduce false positives. For information on generating this file, visit the Fusion Filter GitHub page. Use the ref annot.cdsplus.fa.allvsall.outfmt6.genesym.gz file produced by CTAT. For runs on human genome assemblies GRCh38 and hg19, DRAGEN automatically applies a default file generated using Gencode version 32 annotations for primary chromosomes if no other file is specified using the command line.
--rna-gf-enriched-genes	For RNA enrichment assays, a list of targeted genes specified as one gene-name per line. Only fusion calls involving at least one gene on the list are reported. This option cannot be combined with --rna-gf-enriched-regions.
--rna-gf-enriched-regions	This option is the alternative to --rna-gf-enriched-genes, but the input is provided as a bed-file with region coordinates instead of a gene list. All the genes in the provided GTF that overlap the regions are included. Genes that are extracted are summarized in the *.fusion.enriched_genes.txt output file. This option cannot be combined with --rna-gf-enriched-genes.
--rna-repeat-intervals	BED file that contains a target list of repeat intervals for sensitive fusion detection. Exclusive from --rna-repeat-genes. This option overrides the default files, which contain the genes CIC, DUX4, PSPH, and SEPTIN14 for GRCh38 and hg19 reference genomes.
--rna-repeat-genes	Text file that contains the names or IDs (from annotation GTF file) of targeted repetitive genes for sensitive fusion detection. Exclusive from --rna-repeat-intervals. This option overrides the default BED file.
--enable-variant-annotation	Enable Illumina Annotation Engine (IAE) to report fusion annotations in JSON format. --enable-variant-annotation must be set to true. For more information, see Illumina Annotation Engine on page 653 .
--variant-annotation-assembly	
--variant-annotation-data	

Option	Description
--rna-gf-restrict-genes	When parsing the gene annotations file (GTF/GFF) for use in the DRAGEN Gene Fusion module, you can use this option to restrict the entries of interest to only protein-coding regions. Restricting the GTF to only the protein-coding and lincRNA genes reduces false positive rates in currently studied fusion events. The default value is true.
--rna-gf-merge-calls	If multiple genes overlap a fusion breakpoint, DRAGEN generates and scores a separate fusion candidate for each gene pair overlapping the breakpoint. When reporting such candidates which share the breakpoints when the option is true, DRAGEN merges these into a single row reporting the feature values for the highest scoring passing candidate (or highest scoring failing candidate if no passing candidate is reported). For each breakpoint, in the column #FusionGene, it reports a semi-colon separated list of names of all overlapping genes with a passing candidate. If a mix of passing and failing candidates are reported the same breakpoint pair, genes with only failing candidates are listed in the columns AdditionalGenes1 and AdditionalGenes2. If no passing candidate exists, then then all overlapping genes are reported in the #FusionGene column. The default value is false so that each reported fusion event only has one left and right gene in the fusion, and overlapping genes are output as separate events.
--enable-rna-amplicon	A separate fusion filtering model is trained for RNA amplicon mode. Duplicate removal for fusion supporting reads is disabled for RNA amplicon mode. Both genes are required to be in the list of enriched genes. By default, the DRAGEN fusion caller filters candidates if a transcript overlaps both of the breakpoints (e.g. fusions such as FIP1L1--PDGFRA and GOPC--ROS1). In RNA amplicon mode, such candidates are not filtered. See DRAGEN Amplicon Pipeline for further information. The default is false.
--rna-gf-sv-vcf	Structural variant VCF file output from DRAGEN DNA structural variant caller run in tumor mode. DRAGEN will report SV events matching each fusion candidate and adjust the score based on the present/absence of matching SVs.
--rna-gf-filter-chrm	DRAGEN filters fusion candidates involving chrM/MT with filter MITOCHONDRIAL_GENES if it can autodetect this chromosome. To disable filtering fusions involving mitochondrial genes, set to false. Default is true.

Gene Expression Quantification

The DRAGEN RNA pipeline contains a gene expression quantification module that estimates the expression of each transcript and gene in an RNA data set. The module first internally translates the genomic mapping of each read (read pair) to the corresponding transcript mappings. Then uses an Expectation-Maximization (EM) algorithm to infer the transcript expression values that best match all the observed reads. The EM algorithm can also model and correct for GC-bias in the reported quantification results.

To enable the quantification module, set the `--enable-rna-quantification` option to `true` in your current RNA-seq command-line scripts. Additionally, you must provide a gene annotation file (GTF/GFF) that contains the genomic position of all transcripts to quantify. You can specify the GTF/GFF file using the `-a` or `--annotation-file` option.

Quantification Options

Option	Description
<code>--enable-rna-quantification</code>	If set to true, enables RNA quantification. Requires <code>--enable-rna</code> to be set to true.
<code>--rna-library-type</code>	Specifies the type of RNA-seq library. The following are the available values: <ul style="list-style-type: none"> <code>IU</code>—Paired-end unstranded library. <code>ISR</code>—Paired-end stranded library in which read2 matches the transcript strand (eg, Illumina Stranded Total RNA Prep). <code>ISF</code>—Paired-end stranded library in which read1 matches the transcript strand. <code>U</code>—Single-end unstranded library. <code>SR</code>—Single-end stranded library in which reads are in reverse orientation to the transcript strand (eg, Illumina Stranded Total RNA Prep). <code>SF</code>—Single-end stranded library in which reads match the transcript strand. <code>A</code>—DRAGEN examines the first reads pairs in the data set to automatically detect the correct library type. Autodetect is the default value.
<code>--rna-quantification-gc-bias</code>	GC bias correction estimates the effect of transcript %GC on sequencing coverage and accounts for the effect when estimating expression. To disable GC bias correction, set to false.

Option	Description
--rna-quantification-fld-max	Use these options to specify the insert size distribution of the RNA-seq library for single-end runs. These options are relevant for GC bias correction. The defaults are 250 +- 25. The maximum allowed value is 1000. To improve accuracy, modify the values to match your library.
--rna-quantification-fld-mean	
--rna-quantification-fld-sd	

Quantification Outputs

Transcript quantification results are reported in the `<outputPrefix>.quant.sf` text file. The file lists results for each transcript. You can use the output file as input for differential gene expression using tools such as tximport and DESeq2.

The following is an example of the file contents:

Name	Length	EffectiveLength	TPM	NumReads
ENST00000364415.1	116	12.3238	5.2328	1
ENST00000564138.1	2775	2105.58	1.28293	41.8885

Field	Description
Name	The ID of the transcript.
Length	The length of the (spliced) transcript in base pairs.
EffectiveLength	The length as accessible to RNA-seq, accounting for insert-size and edge effects.
TPM	Transcripts per Million (TPM) represents the expression of the transcript when normalized for transcript length and sequencing depth.
NumReads	The estimated number of reads from the transcript. The values are not normalized.

The gene expression quantification module also outputs the following files. For information on the metrics included, refer to [Quantification and RNA QC Metrics on page 562](#).

- `<outputPrefix>.quant.genes.sf`—Contains quantification results at the gene level. The results are produced by summing together all transcripts with the same geneID in the annotation file (GTF). Length and EffectiveLength are the expression-weighted means of the individual transcripts in the gene.
- `<outputPrefix>.quant.metrics.csv`—Summary statistics relevant to RNA transcripts and quantification. Refer to [Quantification and RNA QC Metrics on page 562](#).

- <outputPrefix>.quant.transcript_fragment_lengths.txt—Full fragment length distribution of reads mapped to transcripts, output in length- probability pairs of length minimum through >999 bases. Summing the products of the two columns will yield the average fragment length.
- <outputPrefix>.quant.transcript_coverage.txt—Measures coverage uniformity with a normalized average of 5' to 3' coverage pattern along transcripts in increments of 1%. A summation of the 100 coverage bins should yield 100%.
- <outputPrefix>.SJ.saturation.txt—Measures sequencing saturation of the library, including the number of unique splice junctions observed as a function of reads processed.

Quantification and RNA QC Metrics

The RNA Quantification module outputs metrics related to the gene expression results and more general RNA QC metrics that rely on transcript-level analysis.

A summary of the metrics is output to the <outputPrefix>.quant.metrics.csv file.

Metric	Description
Library orientation	Library orientation of the RNA-Seq reads relative to the original transcripts. The library orientation can be automatically detected, or can be explicitly provided. See Quantification Options for more information.
Total Genes	Total number of genes from the gene annotation (GTF/GFF) input used for analysis.
Coding Genes	Number of coding genes from the gene annotation (GTF/GFF) excluding pseudo-genes and biotypes which are noncoding.
Total Transcripts	Number of transcripts from the gene annotation file (GTF/GFF) input used for analysis.
Median transcript CV coverage	Median coefficient of variation (CV), or stdev divided by mean coverage, of the 1000 most highly expressed transcripts. This metric measures uniformity of RNA-Seq read coverage.
Median 5' coverage bias	Median 5 prime bias of the 1000 most highly expressed transcripts, calculated per transcript as mean coverage of the 5'-most 100 bases divided by the mean coverage of the whole transcript.

Metric	Description
Median 3' coverage bias	Median 3 prime bias of the 1000 most highly expressed transcripts, calculated per transcript as mean coverage of the 3'-most 100 bases divided by the mean coverage of the whole transcript.
Transcript fragments	Number of fragments (read pairs) mapped to one or more annotated transcripts in the forward or reverse transcript sense.
Strand mismatched fragments	Number of read pairs that do not match the expected strand of the transcript in the case of stranded library orientation. If this is reported, then Forward transcript fragments will not be reported.
Forward transcript fragments	Number of read pairs that match transcripts on the forward strand in the case of unstranded library orientation. The percent column shows the fraction of forward fragments as compared to the Transcript fragments. The number of reverse fragments may be computed as Transcript fragments - Forward transcript fragments.
Ambiguous strand fragments	Read pairs that match transcripts in both forward and reverse orientation.
Intron fragments	Read pairs that overlap with a gene, but do not overlap with an exon.
Intergenic fragments	Read pairs that do not overlap with any gene.
Unknown transcript fragments	Read pairs that overlap with an exon of a gene, but do not match any transcript (mismatched splice sites).
Number of genes with coverage > 1x,10x,30x,100x	The count of the number of genes where the most highly expressed transcript has average coverage greater than 1x, 10x, 20x, and 100x .
Fold coverage of all exons	The average sequencing coverage across all annotated exons, determined using the most highly expressed transcript for each gene.
Fold coverage of coding exons	The average sequencing coverage across only exons within coding genes, determined using the most highly expressed transcript for each gene.
Fold coverage of introns	The average sequencing coverage across detected introns.
Fold coverage of intergenic regions	The average sequencing coverage across areas detected outside annotated genes.

RNA Variant Calling

DRAGEN RNA variant calling uses the DRAGEN Somatic Small Variant Caller to call SNVs and indels. DRAGEN uses somatic variant calling to account for nongermline variant allele frequencies in RNA-Seq data caused by differential expression. To perform variant calling, DRAGEN uses a probability model that weighs the evidence of a real variant against evidence for various noise models. If the quality score for a variant exceeds a certain threshold, then the variant is reported in the output VCF with the PASS label. DRAGEN also applies filters, such as weak_evidence and base_quality, that might indicate if the variant does not reach the thresholds required to qualify as a passing call. For more information on DRAGEN DNA somatic variant calling, see [Somatic Mode on page 126](#).

You can also use force genotyping (ForceGT) with RNA variant calling. You can input a VCF that contains variants of interest, and the output VCF will contain all variants from the input with annotation. ForceGT might be unable to accurately call complex variants or variants with long deletions (> 50 bp). Complex variants are variants that require more than one substitution, insertion, or deletion event to transform the REF allele into the ALT allele.

Input Options

You can use a FASTQ, BAM, or CRAM file as input. Make sure to mark the input file as a tumor file to enable DRAGEN somatic variant calling. Optionally, you can provide a GTF annotation file for more accurate split junction mapping.

Use the following command line options for FASTQ input files.

```
--tumor-fastq1=<fastq1_file> \
--tumor-fastq2=<fastq2_file> \
--RGID=<read_group_id> \
--RGSM=<read_group_sample_name> \
```

Use the following command line options for a list of FASTQ input files.

```
--tumor-fastq-list=<fq_list_file> \
--tumor-fastq-list-sample-id=<sample_id>
```

Use the following command line options for a BAM input file.

```
--tumor-bam-input=<bam_file> \
--enable-map-align=false \
--enable-sort=false \
--enable-duplicate-marking=false
```

Run RNA Variant Calling

To enable RNA variant calling, set `--enable-rna` and `--enable-variant-caller` to true. To enable ForceGT, use `--vc-forcegt-vcf <forcegt_vcf_file>`.

RNA variant calling outputs a VCF file that includes PASS variants and variants that did not pass, due to filters or weak evidence. For more information on filters and additional command line options, see [Somatic Mode on page 126](#).

The following is an example RNA variant calling command line.

```
dragen \
--fastq-file1=<fastq1_file> \
--fastq-file2=<fastq2_file> \
--RGID=<read_group_id> \
--RGSM=<read_group_sample_name> \
--enable-duplicate-marking=true \
--dupmark-version=hash \
--enable-rna=true \
--enable-variant-caller=true \
--ref-dir=<ref_hashtable_dir> \
--output-directory=<output_dir> \
--output-file-prefix=<output_prefix> \
--annotation-file=<gtf_annotation_file> \
--vc-forcegt-vcf=<forcegt_vcf_file>
```

Running RNA Variant Calling with other RNA workflows (eg RNA quantification or RNA fusion calling)

RNA quantification and/or fusion calling can be done with RNA variant calling by including the following options:

`--enable-rna-quantification=true` for RNA quantification

and/or

`--enable-rna-gene-fusion=true` for RNA gene fusions along with the `enable-variant-caller=true` option.

For more information and options related to RNA quantification and fusion calling, see those sections of the user guide.

DRAGEN Single-Cell Pipeline

Multiomics Pipeline

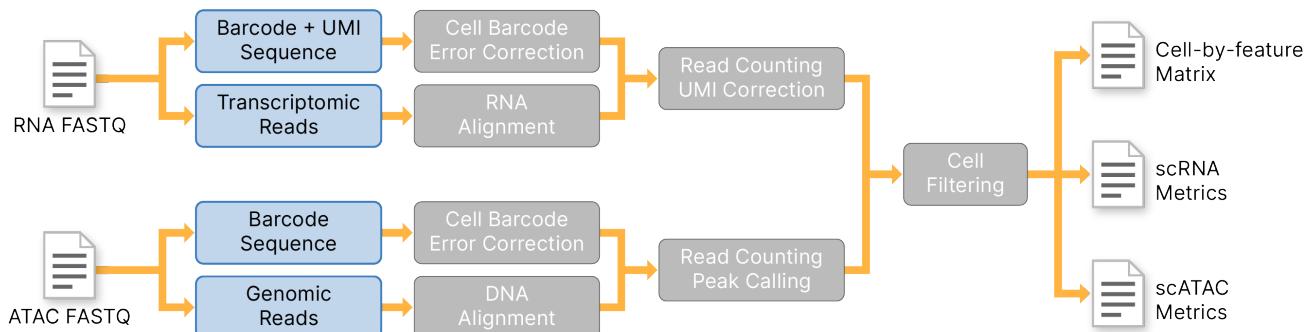
The DRAGEN Single-Cell Multiomics (scRNA + scATAC) Pipeline can process data sets from single-cell RNA-Seq and ATAC-seq reads to a cell-by-feature count matrix.

The pipeline is compatible with library designs that have:

- Single-cell RNA: One read in a fragment match to a transcript and the other containing a cell-barcode and UMI
- Single-cell ATAC: Two reads matching to the genome and the third one containing cell-barcode

The pipeline includes the following functions:

- Alignment
 - RNA-Seq (splice-aware) alignment and matching to annotated genes for the transcript reads
 - ATAC-seq alignment
- Cell-barcode (both RNA- and ATAC-seq) and UMI (RNA-Seq only) error correction for the barcode read
- Read counting
 - UMI counting per cell and gene to measure gene expression
 - Fragment counting per cell and peak to measure chromatin accessibility
- Sparse matrix output and QC metrics



Input Files

Alignment Reference

A standard DRAGEN reference hashtable with both DNA and RNA capability is required for the Single-Cell Multiomics Pipeline. Building a reference hashtable using `--ht-build-rna-hashtable=true` should satisfy this requirement. See the section [Prepare a Reference Genome](#) for more details.

The pipeline also requires a gene annotation file in GTF format, provided with the `--annotation (-a)` option.

Read Input

Because the multiomics workflow assumes at least one pair of single-cell RNA FASTQ files and one triple of single-cell ATAC FASTQ files, the only method to provide read input to DRAGEN is through the `fastq-list` file mechanism. A multiomics `fastq-list` file is a CSV file with the following mandatory columns:

Column	Description
Lane	Sequencing lane
RGID	Read group ID
RGSM	Read group sample
RGLB	Read group library
Read1File	Read 1 FASTQ file
Read2File	Read 2 FASTQ file
UmiFile	FASTQ file with cell-barcodes and/or UMIs
InputGroup	Input group: either scATAC or scRNA (not case sensitive)

Entries in a `fastq-list` file corresponding to single-cell RNA analysis must have Read 2 FASTQ files in the `UmiFile` column and the `Read2File` column entries must be left empty. An example is shown below:

```
Lane,RGID,RGSM,RGLB,Read1File,Read2File,UmiFile,InputGroup
1,rna,sample1,Illumina,scRNA.R1.fastq.gz,,scRNA.R2.fastq.gz,SCRNA
2,atac,sample1,Illumina,scATAC.R1.fastq.gz,scATAC.R2.fastq.gz,scATAC.R3.fastq.gz,SCATAC
```

Settings

To use the multiomics workflow, enter `--enable-rna=true --enable-single-cell-rna=true --enable-single-cell-atac=true`.

Barcode Position

By default the multiomics workflow assumes that the overall barcode/UMI sequence is made up of a single-cell barcode (possibly split into multiple blocks) and a single UMI. Enter the following command to identify the location of the single-cell barcode and single-cell UMI in the barcode read:

```
--scrna-barcode-position <blockPos>[+<blockPos>+<blockPos>...][(:-) | (:+)]
--scrna-umi-position <blockPos>
--scatac-barcode-position <blockPos>[+<blockPos>+<blockPos>...][(:-) | (:+)]
```

For more details, please consult with the corresponding section of scRNA and scATAC workflows' documentation.

Known Barcode Lists

Barcode lists are mandatory for both scRNA and scATAC reads. You need to provide the list of cell barcode sequences using the following command:

```
--scrna-barcode-sequence-list </path/to/scRNABarcodeAllowlist.txt>
--scatac-barcode-sequence-list </path/to/scATACBarcodeAllowlist.txt>
```

The files must contain one possible cell barcode sequence per line. You can compress the file with gzip (*.txt.gz). During cell-barcode error correction any observed barcodes that do not match a sequence specified in the file are considered errors. If possible, the barcodes are corrected to a similar allowed sequence. If the barcodes cannot be corrected, they are filtered out.

Cell Filtering

For each individual modality (either scRNA or scATAC), DRAGEN uses a threshold on the total count of unique UMIs (or reads) per cell barcode, to determine which barcodes are likely to correspond to single-cells in the original sample, instead of background noise. The threshold is determined based on the distribution of counts along barcodes and on the expected number of true cells in the sample. For more information, see the corresponding section in scATAC/scRNA documentation.

After count thresholds in each individual modality is computed, DRAGEN performs a joint cell filtering step. Each cell barcode is represented in a 2-D space with coordinates computed as the total UMI count across genes and the total fragment count across peaks. Initially, a cell-barcode is considered as passing the joint filter if it is passing the filter in each individual modality. DRAGEN then groups all cell barcodes in two categories: those passing both individual modality filters and the rest of cell barcodes. A k-means algorithm with 2 clusters is run and the filtering status of each cell barcode is refined.

Command-line Example

The following is an example command line to run the DRAGEN Single Cell Multiomics Pipeline.

```
dragen \
--enable-rna=true \
--enable-single-cell-atac=true \
-r hg19.fa.default \
--ht-alt-aware-validate=false \
--fastq-list=multiomics_fastq_list.csv \
--umi-source=umifile \
--output-dir=multiomics_output \
--output-file-prefix=sample1 \
```

```
--scatac-barcode-position=0_15 \
--scrna-barcode-position=0_15 \
--scrna-umi-position=16_25 \
--single-cell-threshold=ratio \
--enable-single-cell-rna=true \
-a gencode.v32.primary_assembly.annotation.filtered.gtf \
--scrna-barcode-sequence-whitelist=737K-arc-v1-gex.gz \
--scatac-barcode-sequence-whitelist=737K-arc-v1-atac.gz
```

Outputs

Single-cell ATAC outputs are found in the standard DRAGEN output location using the prefix scATAC.

Counts

The following three files provide information per-cell gene expression level in matrix market (*.mtx) format:

Option	Description
<prefix>.multiomics.matrix.mtx.gz	Count of unique UMIs or fragments for each cell/feature pair in sparse matrix format.
<prefix>.multiomics.barcodes.tsv.gz	Cell-barcode sequence for each cell from the matrix. This includes all cell-barcodes.
<prefix>.multiomics.features.tsv.gz	Feature name and ID for each feature in the matrix.

The subset of barcodes corresponding to passing cells can be found under the Filter column in <prefix>.multiomics.barcodeSummary.tsv indicated by values PASS and FAIL.

The output includes filtered matrix files which only include the per-cell feature count level for the filtered cells in matrix market (*.mtx) format. The multiomics.features.tsv.gz file is common for the unfiltered and filtered matrices:

Option	Description
<prefix>.multiomics.filtered.matrix.mtx.gz	Count of unique UMIs for each filtered cell/feature pair in sparse matrix format.
<prefix>.multiomics.filtered.barcodes.tsv.gz	Cell-barcode sequence for each filtered cell from the matrix.

Loading output in a dense matrix

Loading the matrix in a dense dataframe using python allows you to create an output in a readable format. Illumina recommends a sparse representation of the matrix due to the significant usage of memory and disk space of dense matrices. Several tools are available to work efficiently with "sparse" representations of single cell matrices. Illumina tested the loading the matrix in python 3.10.0 using scanpy 1.9.3 and pandas 1.5.3 tools.

Follow the steps below:

1. Enter the following command to install the required libraries:

```
> pip install -U scanpy pandas
```

2. Use the following python commands to load the matrix in dense representation:

```
# import libraries
import pandas as pd
import scanpy as sc
# define path to input files
matrix_path = "path/to/matrix.mtx.gz"
features_path = "path/to/features.tsv.gz"
barcodes_path = "path/to/barcodes.tsv.gz"
# load matrix through scanpy
adata = sc.read_mtx(matrix_path).T
adata.var_names = pd.read_csv(features_path, sep="\t", header=None,
compression="gzip") [1]
adata.obs_names = pd.read_csv(barcodes_path, sep="\t", header=None,
compression="gzip") [0]
# convert scanpy internal format (AnnData) to dense pandas DataFrame
df = pd.DataFrame(adata.X.todense(), index=adata.obs_names,
columns=adata.var_names)
# save it as CSV file
df.to_csv("output_matrix.csv")
```

The matrix can be saved through different output formats (eg, CSV), although it is not recommended due to high disk usage.

Alignments

DRAGEN Single-Cell Multiomic outputs two BAM files sorted by coordinate - one with suffix `scRNA.bam` and one with suffix `scATAC.bam`. Consult the corresponding section, scATAC or scRNA. of the user guide for more details.

Because the DRAGEN aligner processes RNA and ATAC reads, there will be a separate mapper metrics summary for each modality when running in multiomic mode. The RGID field will be either RNA or ATC based on the read. There is not a common map/align metric summary for all input reads because it would merge the two modalities. The <prefix>.mapping_metrics.csv file will also reflect the modality.

The following is an example of the alignment metrics printed at the end of a DRAGEN Single-Cell Multiomic run:

```
MAPPING/ALIGNING PER RG rna Total reads in RG 52319680 100.00
MAPPING/ALIGNING PER RG rna Number of duplicate marked reads 0 0.00
MAPPING/ALIGNING PER RG rna Number of duplicate marked and mate reads removed
NA
MAPPING/ALIGNING PER RG rna Number of unique reads (excl. duplicate marked
reads) 52319680 100.00
...
MAPPING/ALIGNING PER RG atac Total reads in RG 203484254 100.00
MAPPING/ALIGNING PER RG atac Number of duplicate marked reads 0 0.00
MAPPING/ALIGNING PER RG atac Number of duplicate marked and mate reads removed
NA
MAPPING/ALIGNING PER RG atac Number of unique reads (excl. duplicate marked
reads) 203484254 100.00
...
```

Overall Metrics

The <prefix>.multiomics.metrics.csv file contains per sample scATAC and scRNA metrics. For more details, please consult with the corresponding section from scATAC/scRNA user guide.

Here is an example of how a <prefix>.multiomics.metrics.csv file can look like:

```
SINGLE-CELL ATAC METRICS,lib1,Invalid barcode fragments,0
SINGLE-CELL ATAC METRICS,lib1,Error free cell-barcode,308656
SINGLE-CELL ATAC METRICS,lib1,Error corrected cell-barcode,251364
...
SINGLE-CELL RNA METRICS,lib1,Median genes per cell,1456
SINGLE-CELL RNA METRICS,lib1,Total genes detected,14128
SINGLE-CELL METRICS,lib1,Passing cells,1920
```

Per-Cell Metrics

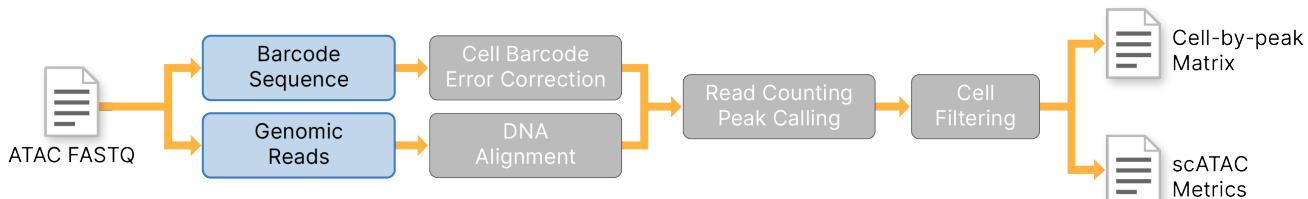
The `<prefix>.multiomics.barcodeSummary.tsv` contains summary statistics for each unique cell-barcode per cell after error correction. Here is an example of how a `<prefix>.multiomics.barcodeSummary.tsv` file can look like:

ID	Barcode	BarcodeScAtac	BarcodeScRna	UniqueFragments	TotalFragments		
PeakFragments	NonPrimaryContigFragments	ChimericFragments	LowMapqFragments	Mitochondria			
1Fragments	Peaks	TotalReads	GeneReads	UMIs	Genes	MitochondrialReads	Filter
1	AAACAAGCAAACAAAG	AAACAAGCAAACAAAG	GGCTAGTGTTCGGTAA	22	48		
13	17	9	59	0	0	2	7
2	AAACAAGCAAACCAGC	AAACAAGCAAACCAGC	GGCTAGTGTACTGAAT	7	7		
3	0	3	8	0	0	0	0
							LOW

ATAC Pipeline

The DRAGEN Single-Cell ATAC (scATAC) Pipeline can process single-cell ATAC-seq data sets from reads to a cell-by-peak read count matrix. The pipeline includes the following functions:

- ATAC-seq alignment
- Cell-barcode error correction for the barcode read
- Chromatin accessibility peak calling
- Fragment counting per cell and peak to measure chromatin accessibility
- Sparse matrix output and QC metrics



The functionality and options related to alignment and gene annotation are identical to DNA pipelines. For information, see DRAGEN DNA Pipeline.

Input Files

Alignment Reference

Use a standard DRAGEN DNA reference genome or hashtable for the scATAC Pipeline.

Read Input

The DRAGEN scATAC Pipeline requires both the genomic sequence and the barcode sequence for each fragment (read) as input. The genomic sequence is aligned to the reference genome to determine the expressed gene, the single-cell barcode sequence is used to identify the unique cell. When starting from FASTQ, you can either include the UMI in the read name or provide separate cell-barcode FASTQ files.

UMI in Read Name

Provide the genomic reads as a paired-end FASTQ files with the Barcode sequence in the eighth field of the read-name line. Separate sequences using a colon. The following example uses read2 (sample.R2.fastq.gz) as the genomic read.

In the example, the GAAACTCGTTCAGCGC sequence is the barcode read and the ACAG... sequence is the genomic read.

These FASTQ files can be generated by bclConvert and bcl2fastq using the UMI settings to define the single-cell barcode read. If using bclConvert, enter the barcode information using the `OverrideCycles1` setting. For more information, see the *BCL Convert Software Guide* (document # 100000094725).

i bclConvert refers to the entire single-cell barcode+UMI sequence as UMI.

Enter the following command line option to use the generated FASTQ files from bclConvert:

```
dragen -l <file name> --umi-source=qname
```

The option is also compatible with the `--fastq-list` input options and with read input from BAM files.

Single-cell ATAC Fastq-list Files

A single-cell ATAC fastq-list file is a CSV file with the following mandatory columns:

Column	Description
Lane	Sequencing lane
RGID	Read group ID

Column	Description
RGSM	Read group sample
RGLB	Read group library
Read1File	Read 1 FASTQ file
Read2File	Read 2 FASTQ file
UmiFile	Read 3 FASTQ file (FASTQ file with cell-barcodes)

See the following example:

```
Lane,RGID,RGSM,RGLB,Read1File,Read2File,UmiFile
1,atac,sample1,illumina1,scATAC1.R1.fastq.gz,scATAC1.R2.fastq.gz,scATAC1.R3.fastq.gz
2,atac,sample1,illumina2,scATAC2.R1.fastq.gz,scATAC2.R2.fastq.gz,scATAC2.R3.fastq.gz
```

Separate UMI FASTQ Files

An alternative option is to provide the genomic and barcode sequences as three separate FASTQ files. Two files contain only the genomic reads and one contains the corresponding barcode-reads in the same order. This file is similar to how read-pairs are normally handled. If using separate UMI files, the sequencing system run setup and bclConvert are not aware of the UMI and treat it as a normal read sequence by default.

Enter the following command line option to use the separate UMI FASTQ files:

```
dragen -1 <file name 1> -2 <file name 2> --umi-fastq=<file name 3> --umi-source=fastq
```

Use the following steps to use this method with multiple FASTQ files:

1. Enter the barcode FASTQ files as read1 in the `fastq-list` file, and then enter the genomic read FASTQ files matching the default `fastq_list.csv` generated by bclConvert as read2 and umifile.
2. Enter the following command:

```
dragen --fastq-list fastq_list.csv --umi-source=umifile
```

Using Multiple Libraries

The scATAC pipeline can process a single biological sample per DRAGEN run. To process multiple single-cell libraries together, split the single sample into multiple single-cell libraries with a unique set of cells in each DRAGEN keeps the cells (barcodes and UMIs) from each library separate and provides merged outputs across all. Read groups are used to specify the library for each FASTQ file using the RGLB attribute.

Settings

To use the scATAC workflow, enter `--enable-single-cell-atac=true`. This section includes information on additional scATAC settings.

Barcode Position

By default the scATAC workflow assumes that the overall barcode sequence is made up of a single-cell barcode (possibly split into multiple blocks). Enter the following command to identify the location of the single-cell barcode:

```
--scatac-barcode-position <blockPos>[+<blockPos>+<blockPos>...][(:-) | (:+)]
```

`blockPos` describes the offset of the first and last inclusive base of the block and is formatted as `<startPos>_<endPos>`. For example, for a library with a 16 bp cell-barcode, enter: `--scatac-barcode-position 0_15`. For a library with the cell-barcode split into three blocks of 9 bp separated by fixed linker sequences and an 8 bp UMI, enter: `--scatac-barcode-position=0_8+21_29+43_51`. By default, the barcode position is assumed to be indicated on the forward strand. To explicitly specify the forward strand, enter: `--scatac-barcode-position 0_15:+` or `--scatac-barcode-position=0_8+21_29+43_51:+`, to explicitly specify the reverse strand, enter: `--scatac-barcode-position 0_15:-` or `--scatac-barcode-position=0_8+21_29+43_51:-`.

Known Barcode Lists

You can provide a list of cell barcode sequences to include using the following command:

```
--single-cell-barcode-sequence-whitelist </path/to/barcodeAllowlist.txt>
```

If the `--scatac-barcode-position` parameter is not split into multiple blocks the file must contain one possible cell barcode sequence per line. Refer to the previous Barcode Position section for more information. When the barcode position is split into multiple blocks the following requirements are needed:

- The file must contain a list composed by multiple sections, one for each block.
- Each section must indicate the possible cell barcode block sequences for the corresponding block.
- Each section should start with a line with prefix `#-`, e.g.

Refer to the following example:

```
#-Block1
<barcode_block_1_sequence_1>
<barcode_block_1_sequence_2>
#-Block2
<barcode_block_2_sequence_1>
<barcode_block_2_sequence_2>
```

...

The input file might be compressed with gzip (*.txt.gz).

During cell-barcode error correction any observed barcodes that do not match a sequence specified in the file are considered errors. If possible, the barcodes are corrected to a similar allowed sequence. See [Barcode Error Correction](#) for more information. If the barcodes cannot be corrected, they are filtered out.

Cell Filtering

DRAGEN uses a threshold on the total count of unique reads per cell barcode, to determine which barcodes are likely to correspond to single-cells in the original sample, instead of background noise. The threshold is determined based on the distribution of counts along barcodes and on the expected number of true cells in the sample. For more information on how cell filtering is performed, see [Cell Filtering](#).

- *single-cell-number-cells*—[Optional] Set the expected number of cells. The default is 3000. Adjust only if the expected number of cells is so far from the default that DRAGEN does not call the correct cell filtering threshold automatically.
- *single-cell-threshold*—Specify the method for determining the count threshold value. The available values are `fixed`, `ratio`, or `inflection`.
 - If using `ratio`, DRAGEN estimates the expected number of cells as $\max(T_e, T_m)$. T_m is a threshold based on a fraction of the counts seen in most abundant cell-barcodes. T_e is a threshold based on a fraction of the least abundant expected cell.
 - If using `inflection`, DRAGEN estimates the count threshold by analyzing inflection points in the cumulative distribution of counts.
 - If using `fixed`, the count threshold is set to force the expected number of cells (`single-cell-number-cells` option), rather than estimating it from the data. The exact number of passing cells might be slightly larger than the number of requested single-cells because several cells in the tail of the count distribution can have the same count.

To set a specific a fixed number of cells, rather than use the automatically determined threshold, use the following command:

```
--single-cell-threshold=fixed --single-cell-number-cells=X
```

The command forces DRAGEN to select the top X cells and extra cells with the same number of counts of the last selected cell.

Additional Options

The following are additional options you can use to configure the Single-Cell ATAC Pipeline settings.

Option	Description
<code>qc-enable-depth-metrics</code>	Set to <code>false</code> to disable depth metrics for faster run times. The default is <code>true</code> .
<code>scatac-write-fragments</code>	Set to <code>true</code> to write counted fragments to the disk in both <code>tsv <prefix>.scATAC.fragments.tsv</code> and <code>BigWig <prefix>.scATAC.fragments.bigwig</code> formats. The default is <code>false</code> .

Command-line Example

The following is an example command line to run the DRAGEN Single Cell RNA Pipeline.

```
dragen \
--enable-single-cell-atac=true \
--umi-source=fastq \
--scatac-barcode-position 0_15 \
-r reference_genomes/Mus_musculus/mm10/DRAGEN/8 \
-1 lib1_S7_L001_R1_001.fastq.gz \
-2 lib1_S7_L001_R2_001.fastq.gz \
--umi-fastq lib1_S7_L001_R3_001.fastq.gz \
--RGID=1 \
--RGSM=sample1 \
--output-dir=/staging/out \
--output-file-prefix=sample1
```

Outputs

Single-cell ATAC outputs are found in the standard DRAGEN output location using the prefix scATAC.

Counts

The following three files provide information per-cell gene expression level in matrix market (*.mtx) format:

Option	Description
<code><prefix>.scATAC.matrix mtx.gv</code>	Count of unique UMIs for each cell/gene pair in sparse matrix format.
<code><prefix>.scATAC.barcodes tsv.gv</code>	Cell-barcode sequence for each cell from the matrix. This includes all cell-barcodes.
<code><prefix>.scATAC.peaks tsv.gv</code>	Peak name and ID for each peak in the matrix.

The subset of barcodes corresponding to passing cells can be found under the Filter column in <prefix>.scATAC.barcodeSummary.tsv indicated by values PASS and FAIL.

The output includes filtered matrix files which only include the per-cell chromatin accessibility level for the filtered cells in matrix market (*.mtx) format. The scATAC.peaks.tsv.gz file is common for the unfiltered and filtered matrices:

Option	Description
<prefix>.scATAC.filtered.matrix.mtx.gz	Count of unique UMIs for each filtered cell/peak pair in sparse matrix format.
<prefix>.scATAC.filtered.barcodes.tsv.gz	Cell-barcode sequence for each filtered cell from the matrix.

Loading Output in a Dense Matrix

Loading the matrix in a dense dataframe using python allows you to create an output in a readable format. Illumina recommends a sparse representation of the matrix due to the significant usage of memory and disk space of dense matrices. Several tools are available to work efficiently with "sparse" representations of single cell matrices. Illumina tested the loading the matrix in python 3.10.0 using scanpy 1.9.3 and pandas 1.5.3 tools.

Follow the steps below:

- Enter the following command to install the required libraries:

```
> pip install -U scanpy pandas
```

- Use the following python commands to load the matrix in dense representation:

```
# import libraries
import pandas as pd
import scanpy as sc
# define path to input files
matrix_path = "path/to/matrix.mtx.gz"
features_path = "path/to/peaks.tsv.gz"
barcodes_path = "path/to/barcodes.tsv.gz"
# load matrix through scanpy
adata = sc.read_mtx(matrix_path).T
adata.var_names = pd.read_csv(features_path, sep="\t", header=None,
compression="gzip") [1]
adata.obs_names = pd.read_csv(barcodes_path, sep="\t", header=None,
compression="gzip") [0]
# convert scanpy internal format (AnnData) to dense pandas DataFrame
```

```
df = pd.DataFrame(adata.X.todense(), index=adata.obs_names,
columns=adata.var_names)
# save it as CSV file
df.to_csv("output_matrix.csv")
```

The matrix can be saved through different output formats (eg, CSV), although it is not recommended due to high disk usage.

Alignments

Alignments of the genomic reads are sorted by coordinate and output as a BAM file. Each alignment is annotated with an XB tag containing the cell-barcode. The alignments use the original sequences without any errors corrected. Fragments that did not have an associated barcode read, for example fragments trimmed on the input data, do not have XB and tag.

Overall Metrics

The <prefix>.scATAC.metrics.csv file contains per sample scATAC metrics.

Barcode Read Metrics

Metric	Description
Invalid barcode read	Overall barcode sequence failed basic checks. For example, the barcode read was missing or too short.
Error free cell-barcode	Reads with cell-barcode sequences that were not altered during error correction. For example, if the read was an exact match to the allow list.
Error corrected cell-barcode	Reads with cell-barcode sequences successfully corrected to a valid sequence.
Filtered cell-barcode	Reads with cell-barcode sequences that could not be corrected to a valid sequence. For example, the sequence does not match allow list with at most one mismatch.

Genomic Fragment Metrics

Metric	Description
Fragments passing filters	Non-chimeric non-mitochondrial fragments that align to primary contigs with a high mapping quality (greater than 30 by default).

Metric	Description
Non-primary contig fragments	Fragments that align to non-primary contigs (any contigs that are not autosome, X and Y).
Chimeric fragments	Fragments with the two reads aligning to different contigs.
Mitochondrial fragments	Fragments aligning to the mitochondrial contigs.
Low mapping quality fragments	Fragments with the two reads aligning with a mapping quality set to some specific value (default is 30).
Improperly mapped fragments	The two reads in the fragment are not mapped in proper pair (SAM flag "read mapped in proper pair" is set to 0).

Cell Metrics

Metric	Description
Fragment threshold for passing cells	Number of fragments required for a cell-barcode to pass filtering.
Passing cells	Number of cell-barcodes that passed the filters.
Fraction peak fragments in passing cells:	Percentage of counted fragments intersecting peaks assigned to cells that passed the filters.
Fraction fragments in passing cells	Percentage of all counted fragments assigned to cells that passed the filters.
Median fragments per cells	Total counted fragments per cell that passed the filters.
Median peaks per cells	Peaks with at least one fragment per cell that passed the filters.
Total peaks detected	Peaks with at least one fragment in at least one cell that passed the filters.

Per-Cell Metrics

The `<prefix>.scATAC.barcodeSummary.tsv` contains summary statistics for each unique cell-barcode per cell after error correction.

Metric	Description
ID	Unique numeric ID for the cell-barcode.
Barcode	The cell-barcode sequence.
TotalFragments	Total fragments with the cell-barcode sequence.
UniqueFragments	Unique fragments counted towards a peak.

Metric	Description
NonPrimaryContigFragments	Unique non-primary contig framgnets.
ChimericFragments	Unique chimeric fragments.
LowMapqFragments	Unique low mapping quality fragments.
MitochondrialFragments	Unique fragments mapped to mitochondrial genome.
Peaks	Unique peaks detected
Filter	The following are the available filter values: <ul style="list-style-type: none">• PASS - Cell-barcode passes the filter• LOW - UMI count is below threshold

Barcode Error Correction

Cell-barcode sequences from the input reads are error corrected based on the frequency with which each one is seen and an optional allow list of expected cell-barcode sequences. A cell-barcode sequence is considered a neighbor of another cell-barcode if there is at most one mismatch. A cell-barcode sequence is corrected to its neighbor in the following circumstances. When corrected, all reads with the cell-barcode are assigned instead to the neighboring cell-barcode. The sequence error correction scheme is similar to the directional algorithm described in (Smith, Heger and Sudbery, 2020)¹.

- The neighboring cell-barcode is at least two times more frequent across all input reads.
- The neighboring cell-barcode is on the cell-barcode allow list, but the original cell-barcode is not.

To avoid overcounting cell-barcodes based on sequence errors, cell-barcode error correction is performed among all reads with the same cell-barcode mapping to the same peak region. Cell-barcode sequences that are likely errors of another cell-barcodes are not counted.

¹ Smith,T., Heger, A. and Sudbery, I., 2020. UMI-Tools: Modeling Sequencing Errors In Unique Molecular Identifiers To Improve Quantification Accuracy. [PDF] Cold Spring Harbor Laboratory Press. Available at: <<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5340976/>> [Accessed 1 March 2022].

Peak Calling

DRAGEN calls peaks using an algorithm based on MACS, version 3 (Zhang et al., 2008)¹. To customize the behavior of the peak calling algorithm, modify any of the command line parameters specified in the table below.

Command line parameter	Meaning	Default value
atac-peak-qvalue	Threshold for q-value to call peaks	0.05

Command line parameter	Meaning	Default value
atac-peak-fold-change	Fold change threshold (relative to the background) to call peaks	1.0
atac-peak-min-length	Minimum length of a peak, bp	NA The default value is computed automatically as the mean fragment length.
atac-peak-max-gap	Maximum pileup gap (if the gap is larger - initiate another peak), bp	NA The default value is computed automatically as the mean fragment length.

Alternatively, if fragment counting needs to be performed on a pre-specified set of peaks, provide a peak BED file using command line parameter `atac-peak-bed-file`.

¹ Zhang,Y., Liu, T., Meyer, C.A., Eeckhoute, J., Johnson, D.S., Bernstein, B.E., Nusbaum, C., Myers, R.M., Brown M., Li, W. and Liu, X.S., 2008. Model-based Analysis of ChIP-Seq (MACS). [PDF] Available at: <<https://pubmed.ncbi.nlm.nih.gov/18798982/>> [Accessed 1 March 2022].

Peak Annotation

DRAGEN annotates each peak with respect to a gene symbol as promoter, distal, or intergenic depending on the genomic position of both the peak and the gene. The following rules are used to determine the annotation of a peak:

- If a peak overlaps with promoter region (-1000bp, +100bp) of any transcription start site (TSS), it is annotated as a promoter peak of the gene.
- If a peak is within 200kb of the closest TSS, and if it is not a promoter peak of the gene of the closest TSS, it will be annotated as a distal peak of that gene.
- If a peak overlaps the body of a transcript, and it is not a promoter nor a distal peak of the gene, it will be annotated as a distal peak of that gene with distance set as zero.
- If a peak has not been mapped to any gene at the step, it will be annotated as an intergenic peak without a gene symbol assigned.

To enable peak annotation in DRAGEN scATAC-seq workflow, specify a gene annotation file (GTF) using the option `-a`. Peak annotations are written to a file with name `<prefix>.scATAC.peaks.tsv` and each annotation is represented as a row with the following 6 columns:

- Chromosome number
- Start position
- End position
- Gene symbol
- Distance from peak to gene
- Peak annotation (i.e, promoter, distal, or intergenic).

Transcription Factor Motif Analysis

In this analysis, peaks are matched to a set of known transcription factor (TF) binding sites and for each cell barcode the fragment counts are grouped based on transcription factors their peaks are assigned to. This results in a more compact representation of chromatin accessibility patterns. To enable TF motif analysis, specify a database of position-weight matrices (PWM) corresponding to transcription factor motifs in JASPAR format:

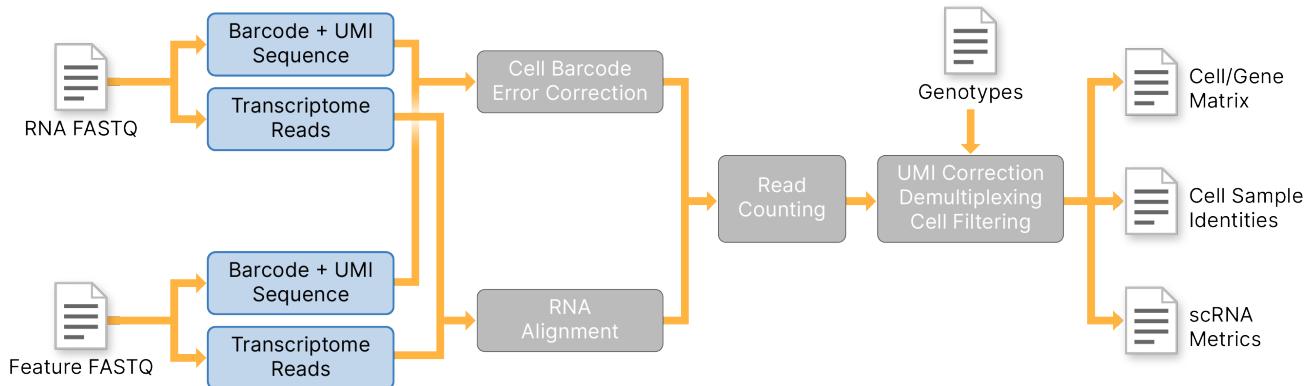
```
--atac-jaspar-database=JASPAR2022_CORE_non-redundant_pfms_jaspar.txt
```

DRAGEN will produce two files `<prefix>.scATAC.motifs.matrix mtx` and `<prefix>.scATAC.tf.motifs.tsv` which combined with file `<prefix>.scATAC.barcodes.tsv` (also used in the cell-by-peak count matrix) represent the cell-by-TF count matrix in matrix market format.

RNA Pipeline

The DRAGEN Single-Cell RNA (scRNA) Pipeline can process multiplexed single-cell RNA-Seq data sets from reads to a cell-by-gene UMI count gene expression matrix. The pipeline is compatible with library designs that have one read in a fragment matched to a transcript and the other containing a cell-barcode and UMI. The pipeline includes the following functions:

- RNA-Seq (splice-aware) alignment and matching to annotated genes for the transcript reads.
- Cell-barcode and UMI error correction for the barcode read.
- UMI counting per cell and gene to measure gene expression.
- Sparse matrix output and QC metrics.
- Feature counting, such as with cell-surface proteins.



The functionality and options related to alignment and gene annotation are identical to the RNA-Seq pipeline. For information, see [DRAGEN RNA Pipeline on page 537](#). Other RNA-Seq modules, such as gene fusion calling or transcript-level gene expression quantification are not supported for Single-Cell RNA.

Input Files

Alignment Reference

Use a standard DRAGEN RNA reference genome or hashtable for the DRAGENscRNA Pipeline. For example, build using `--ht-build-rna-hashtable=true`). The pipeline also requires a gene annotation file in GTF format, provided with the `--annotation (-a)` option.

Read Input

The DRAGEN scRNA Pipeline requires both the transcript sequence and the barcode+UMI sequence for each fragment (read) as input. The transcript sequence is aligned to the genome to determine the expressed gene, the barcode+UMI sequence is split into the single-cell barcode to identify the unique cell, and a single-cell UMI for unique molecule quantification.

When starting from FASTQ, you can either include the UMI in the read name or provide separate UMI FASTQ files.

UMI in Read Name

Provide the transcriptome reads as a single-end FASTQ file with the Barcode+UMI sequence in the eighth field of the read-name line. Separate sequences using a colon. The following example uses `read2 (sample.R2.fastq.gz)` as the transcript read.

```

@D00626:253:CAT5WANXX:4:1302:8433:85343:GAAACTCGTTCAGCGCTCATCTCGGC
ACAGGTCAGCTGGGAGCTTCTGCCCTACTGCCTAGGGACCAACAGGGGCAGGAGGCAGTCACTGACCCGAGAAGTT
GCATCCTGCACAGCTAGAGATC
+
  
```

In the example, the GAA sequence is the barcode+UMI read and the ACAG sequence is the transcriptome read.

These FASTQ files can be generated by bclConvert and bcl2fastq using the UMI settings to define the single-cell barcode+UMI read. If using bclConvert, enter the barcode/UMI information using the `OverrideCycles` setting. For more information, see the *BCL Convert Software Guide* (document # 100000094725).

i bclConvert refers to the entire single-cell barcode+UMI sequence as UMI.

Enter the following command line option to use the generated FASTQ files from bclConvert:

```
dragen -l <file name> --umi-source=qname
```

The option is also compatible with the `--fastq-list` input options and with read input from BAM files.

Single-cell RNA Fastq-list Files

A single-cell RNA fastq-list file is a CSV file with the following mandatory columns:

Column	Description
Lane	Sequencing lane
RGID	Read group ID
RGSM	Read group sample
RGLB	Read group library
Read1File	Read 1 FASTQ file (Usually FASTQ file with reads containing cell-barcodes and UMIs.)
Read2File	Read 2 FASTQ file (Usually transcriptomic reads.)

In this case, DRAGEN needs to accept the `--umi-source=read1`, or `--umi-source=read2` if swapped, command-line option.

For example, a fastq-list file with the following contents must be used in combination with the `--umi-source=read1` option:

```
Lane,RGID,RGSM,RGLB,Read1File,Read2File  
1,rna,sample1,illuminal,scRNA1.R1.fastq.gz,scRNA1.R2.fastq.gz  
2,rna,sample1,illumina2,scRNA2.R1.fastq.qz,scRNA2.R2.fastq.qz
```

Alternatively, the FASTQ file with transcriptomic reads can be specified under the Read1File column and the FASTQ file with cell barcodes and/or UMIs under the UmiFile column.

For example, a fastq-list file with the following contents must be used in combination with the --umi-source=umifile option:

```
Lane,RGID,RGSM,Read1File,UmiFile
1,rna,sample1,illuminal,scRNA1.R2.fastq.gz,scRNA1.R1.fastq.gz
2,rna,sample1,illumina2,scRNA2.R2.fastq.gz,scRNA2.R1.fastq.gz
```

Separate UMI FASTQ Files

An alternative option is to provide the transcript and barcode+UMI sequences as two separate FASTQ files. One file contains only the transcriptome reads and one contains the corresponding barcode-reads in the same order. This file is similar to how read-pairs are normally handled. If using separate UMI files, the sequencing system run setup and bclConvert are not aware of the UMI and treat it as normal read sequence by default.

Enter the following command line option to use the separate UMI FASTQ files:

```
dragen -1 <file name> --umi-fastq=<file name> --umi-source=fastq
```

To use this method with multiple FASTQ files, do as follows.

1. Enter the barcode+UMI FASTQ files as read1 in the fastq-list file, and then enter the transcriptome read FASTQ files matching the default fastq_list.csv generated by bclConvert as read2.
2. Enter the following command:

```
dragen --fastq-list fastq_list.csv --umi-source=read1
```

Using Multiple Libraries

The scRNA pipeline can process a single biological sample per DRAGEN run. To process multiple single-cell libraries together, split the single sample into multiple single-cell libraries with a unique set of cells in each DRAGEN keeps the cells (barcodes and UMIs) from each library separate and provides merged outputs across all. Read groups are used to specify the library for each FASTQ file using the RGLB attribute.

Settings

To use the scRNA workflow, enter --enable-rna=true --enable-single-cell-rna=true. This section includes information on additional scRNA settings.

Barcode Position

By default the scRNA workflow assumes that the overall barcode/UMI sequence is made up of a single-cell barcode (possibly split into multiple blocks) and a single UMI. Enter the following command to define the location of the single-cell barcode and single-cell UMI in the barcode read:

```
--scrna-barcode-position <blockPos>[+<blockPos>+<blockPos>...] [(:-) | (:+)] --
scrna-umi-position <blockPos>
```

`blockPos` describes the offset of the first and last inclusive base of the block and is formatted as `<startPos>_<endPos>`. For example, for a library with a 16 bp cell-barcode followed by a 10 bp UMI, use: `--scrna-barcode-position 0_15 --scrna-umi-position 16_25`. For a library with the cell-barcode split into three blocks of 9 bp separated by fixed linker sequences and an 8 bp UMI, use: `--scrna-barcode-position=0_8+21_29+43_51 --scrna-umi-position=52_59`. By default, the barcode position is assumed to be indicated on the forward strand. To explicitly specify the forward strand, use: `--scrna-barcode-position 0_15:+` or `--scrna-barcode-position=0_8+21_29+43_51:+`. To explicitly specify the reverse strand, use: `--scrna-barcode-position 0_15:-` or `--scrna-barcode-position=0_8+21_29+43_51:-`.

UMI position can also be specified for feature reads, which are reads with a sequence tag specific to a feature (eg, cell-surface protein or antibody). When the feature-specific UMIs are located on Read 2, you can use `--scrna-feature-barcode-r2umi=0_11` to specify a 12 bp feature UMI at the beginning of each feature read.

Specify Barcodes

You can provide a list of cell barcode sequences to include using the following command:

```
--single-cell-barcode-sequence-whitelist </path/to/barcodeAllowlist.txt>
```

In the case where the `--scrna-barcode-position` parameter is not split into multiple blocks (see Barcode Position section) the file must contain one possible cell barcode sequence per line. Differently, when the barcode position is split into multiple blocks, the file must contain a list composed by multiple sections (one for each block): each section must indicate the possible cell barcode block sequences for the corresponding block. Each section should start with a line with prefix `#-`, e.g.:

```
#-Block1
<barcode_block_1_sequence_1>
<barcode_block_1_sequence_2>
...
#-Block2
<barcode_block_2_sequence_1>
<barcode_block_2_sequence_2>
...
```

You can compress the file with gzip (*.txt.gz). During cell-barcode error correction any observed barcodes that do not match a sequence specified in the file are considered errors. If possible, the barcodes are corrected to a similar allowed sequence. See [Barcode Error Correction on page 593](#) for more information. If the barcodes cannot be corrected, they are filtered out.

Cell Filtering

DRAGEN uses a threshold on the number of unique UMIs per cell barcode to determine which barcodes likely correspond to single-cells in the original sample from background noise. The threshold is determined based on the distribution of UMIs per barcode and an expected number of true cells in the sample. For more information on how cell filtering is performed, see [CellFiltering_fDG](#).

- *single-cell-number-cells*—[Optional] Set the expected number of cells. The default is 3000. Adjust only if the expected number of cells is so far from the default that DRAGEN does not call the correct cell filtering threshold automatically.
- *single-cell-threshold*—Specify the method for determining the UMI count threshold value. The available values are `fixed`, `ratio`, or `inflection`.
 - If using `ratio`, DRAGEN estimates the expected number of cells as $\max(T_e, T_m)$. T_m is a threshold based on a fraction of the UMIs seen in most abundant cell-barcodes. T_e is a threshold based on a fraction of the least abundant expected cell.
 - If using `inflection`, DRAGEN determines the threshold using an algorithm based on the inflection point analysis of the number of cumulative UMI counts of the most UMI abundant cells.
 - If using `fixed`, the UMI count threshold is set so that the expected number of cells pass, rather than estimated dynamically. The exact number of passing cells might be slightly larger than the number of single-cells because of the connection between the UMI count and the different cell barcodes.

To set a specific number of cells ahead of time, use the following command:

```
--single-cell-threshold=fixed --single-cell-number-cells=X
```

The command forces the UMI threshold value to pass the top X cells and any extra cells with the same number of unique UMIs.

Additional Options

The following are additional options you can use to configure the Single-Cell RNA Pipeline settings.

Option	Description
<i>rna-library-type</i>	Set the orientation of transcript reads relative to the genomes. Enter <code>SF</code> for forward, <code>SR</code> for reverse, or <code>U</code> for unstranded. The default is <code>SF</code> .

Option	Description
<i>single-cell-count-introns</i>	Include intronic reads in gene expression estimation. The default false.
<i>qc-enable-depth-metrics</i>	Set to <code>false</code> to disable depth metrics for faster run times. The default is true.
<i>bypass-anchor-mapping</i>	Set to <code>true</code> to disable RNA anchor (two-pass) mapping for increased performance. The default false.

Command-line Example

The following is an example command line to run the DRAGEN Single Cell RNA Pipeline.

```
dragen \
--enable-rna=true \
--enable-single-cell-rna=true \
--umi-source=fastq \
--scrna-barcode-position 0_15 \
--scrna-umi-position 16_25 \
-r reference_genomes/Mus_musculus/mm10/DRAGEN/8 \
-a reference_genomes/Mus_musculus/mm10/gtf/gencode.vM23.annotation.gtf.gz \
-l lib1_S7_L001_R2_001.fastq.gz \
--umi-fastq lib1_S7_L001_R1_001.fastq.gz \
--RGID=1 \
--RGSM=sample1 \
--output-dir=/staging/out \
--output-file-prefix=sample1
```

Outputs

Single-cell RNA outputs are found in the standard DRAGEN output location using the prefix scRNA.

Counts

The following three files provide information per-cell gene expression level in matrix market (*.mtx) format:

Option	Description
<code><prefix>.scRNA.matrix.mtx.gz</code>	Count of unique UMIs for each cell/gene pair in sparse matrix format.

Option	Description
<prefix>.scRNA.barcodes.tsv.gz	Cell-barcode sequence for each cell from the matrix. This includes all cell-barcodes.
<prefix>.scRNA.genes.tsv.gz	Gene name and ID for each gene in the matrix.

The subset of barcodes corresponding to passing cells can be found under the Filter column in <prefix>.scRNA.barcodeSummary.tsv indicated by values PASS and FAIL.

The output includes filtered matrix files which only include the per-cell gene expression for the filtered PASS cells in matrix market *.mtx format. The scRNA.genes.tsv.gz files is common for the unfiltered and filtered matrices:

Option	Description
<prefix>.scRNA.filtered.matrix.mtx.gz	Count of unique UMIs for each filtered cell/gene pair in sparse matrix format.
<prefix>.scRNA.filtered.barcodes.tsv.gz	Cell-barcode sequence for each filtered cell from the matrix.

Loading output in a dense matrix

Loading the matrix in a dense dataframe using python allows you to create an output in a readable format. Illumina recommends a sparse representation of the matrix due to the significant usage of memory and disk space of dense matrices. Several tools are available to work efficiently with "sparse" representations of single cell matrices. Illumina tested the loading the matrix in python 3.10.0 using scanpy 1.9.3 and pandas 1.5.3 tools.

Follow the steps below:

- Enter the following command to install the required libraries:

```
> pip install -U scanpy pandas
```

- Use the following python commands to load the matrix in dense representation:

```
# import libraries
import pandas as pd
import scanpy as sc
# define path to input files
matrix_path = "path/to/matrix.mtx.gz"
features_path = "path/to/features.tsv.gz"
barcodes_path = "path/to/barcodes.tsv.gz"
# load matrix through scanpy
adata = sc.read_mtx(matrix_path).T
```

```

adata.var_names = pd.read_csv(features_path, sep="\t", header=None,
compression="gzip") [1]
adata.obs_names = pd.read_csv(barcodes_path, sep="\t", header=None,
compression="gzip") [0]
# convert scanpy internal format (AnnData) to dense pandas DataFrame
df = pd.DataFrame(adata.X.todense(), index=adata.obs_names,
columns=adata.var_names)
# save it as CSV file
df.to_csv("output_matrix.csv")

```

The matrix can be saved through different output formats (eg, CSV), although it is not recommended due to high disk usage.

Alignments

Alignments of the transcript reads are sorted by coordinate and output as a BAM file. Each alignment is annotated with an `XB` tag containing the cell barcode and an `RX` tag containing the UMI. The alignments use the original sequences without any errors corrected. Fragments that do not have an associated barcode read, for example fragments trimmed on the input data, do not have XB and RX tags.

Overall Metrics

The `<prefix>.scRNA.metrics.csv` file contains per sample scRNA metrics.

Barcode Read Metrics

Metric	Description
Invalid barcode read	Overall barcode sequence (cell barcode + UMI) failed basic checks. For example, the barcode read was missing or too short.
Error free cell-barcode	Reads with cell-barcode sequences that were not altered during error correction. For example, if the read was an exact match to the allow list.
Allow listed cell-barcode	Reads with cell-barcode sequences successfully corrected to a valid sequence.
Filtered cell-barcode	Reads with cell-barcode sequences that could not be corrected to a valid sequence. For example, the sequence does not match allow list with at most one mismatch.

Transcript Read Metrics

Metric	Description
Unique exon match	Reads with valid cell-barcode and UMI that match a unique gene.
Unique intron match	Reads do not match exons, but introns of exactly one gene. For example, if using the command <code>--single-cell-count-introns=true</code> .
Ambiguous match	Reads match to multiple genes.
Wrong strand	Reads overlap a gene on the opposite strand defined by library type.
Mitochondrial reads	Reads map to the mitochondrial example, if there is a matching gene.
No gene	Reads do not match to any gene. Includes intronic reads unless using <code>--single-cell-count-introns=true</code> .
Filtered multimapper	Reads excluded due to multiple alignment positions in the genome.
Feature reads	Reads matching to features, when using feature counting.

UMI Count Metrics

Metric	Description
Total counted reads	Reads with valid cell-barcode and UMI that match a unique gene.
Reads with error-corrected UMI	Counted reads where the UMI was error-corrected to match another similar UMI sequence.
Reads with invalid UMI	Reads that were not counted due to invalid UMI sequence. For example, pure homopolymer reads or reads containing Ns.
Sequencing saturation	Fraction of reads with duplicate UMIs. $1 - (\text{UMIs} / \text{Reads})$.
Unique cell-barcodes	Overall number of unique cell-barcode sequences in counted reads only.
Unique UMIs	Overall number of unique cell-barcode and UMI combinations counted.

Cell Metrics

Metric	Description
UMI threshold for passing cells	Number of UMIs required for a cell-barcode to pass filtering.
Passing cells	Number of cell-barcodes that passed the filters.
Fraction genic reads in cells	Counted reads assigned to cells that passed the filters.
Fraction reads putative cells	All counted reads assigned to cells that passed the filters.
Median reads per cells	Total counted reads per cell that passed the filters.
Median UMIs per cells	Total counted UMIs per cell that passed the filters.
Median genes per cells	Genes with at least one UMI per cell that passed the filters.
Total genes detected	Genes with at least one UMI in at least one cell that passed the filters.

Per-Cell Metrics

The `<prefix>.scRNA.barcodeSummary.tsv` contains summary statistics for each unique cell-barcode per cell after error correction.

Metric	Description
ID	Unique numeric ID for the cell-barcode. The ID matches the line in UMI count matrix (*.mtx) output.
Barcode	The cell-barcode sequence.
TotalReads	Total reads with the cell-barcode sequence. This includes error corrected reads.
GeneReads	Reads counted towards a gene.
UMIs	Total number of UMIs in counted reads.
Genes	Unique genes detected.
MitochondrialReads	Reads mapped to mitochondrial genome.
Filter	The following are the available filter values: <ul style="list-style-type: none"> PASS—Cell-barcode passes the filter. LOW—UMI count is below the threshold.

Barcode Error Correction

Cell-barcode sequences from the input reads are error corrected based on their frequency, and optionally through a list of expected cell-barcode sequences. A cell-barcode sequence is corrected into another cell-barcode sequence if they differ only by one base (1 Hamming distance) and:

- Either the corrected cell-barcode is at least two times more frequent across all input reads
- Or
- The corrected cell-barcode is on the list of expected cell-barcode sequences, but the original cell-barcode is not

When corrected, all the original cell-barcode reads are assigned to the corrected cell-barcode. The sequence error correction scheme is similar to the directional algorithm described in (Smith, Heger and Sudbery, 2020)¹.

To avoid overcounting UMIs based on sequence errors, UMI error correction is performed among all reads with the same cell-barcode mapping to the same gene. UMI sequences that are likely errors of another UMI are not counted.

¹ Smith,T., Heger, A. and Sudbery, I., 2020. UMI-Tools: Modeling Sequencing Errors In Unique Molecular Identifiers To Improve Quantification Accuracy. [PDF] Cold Spring Harbor Laboratory Press. Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5340976>.

Genotype Sample Demultiplexing

DRAGEN implements several strategies for demultiplexing of data sets that represent mixtures of cells from different individuals, such as cells pooled in one library prep or microfluidic run. Two of these strategies include a genotype-based and genotype-free demultiplexing. In genotype demultiplexing methods, DRAGEN can assign sample identity to cells based on alleles observed in reads in each cell. DRAGEN only accounts for SNVs. DRAGEN flags any doublets, such as droplets that contain multiple cells from different individuals.

To use genotype-based sample demultiplexing, you must provide a VCF file with genotypes for each sample in the data set. To use genotype-free sample demultiplexing, you must provide a VCF file with a set of external samples preferably coming from a population with the same genetic background. The `GT` field represents the sample genotypes.

For information on the cell-hashing demultiplexing method, see [Cell-Hashing on page 597](#)

Command-Line Options

You can use the following command line options for scRNA demultiplexing.

Option	Description
<code>--scrna-demux-sample-vcf</code>	If using genotype-based sample demultiplexing, specify the VCF file that contains the sample genotypes.
<code>--scrna-demux-reference-vcf</code>	If using genotype-free sample demultiplexing, specify the VCF file that contains the genotypes of a population with a similar genetic background to the samples you are using.

Option	Description
--scRNA-demux-detect-doublets	Enable the doublet detection in genotype-based sample demultiplexing. The default value is <code>false</code> .
--scRNA-demux-number-sample	The number of samples you are using. This option is only applicable when using an external VCF reference specified with the <code>scRNA-demux-reference-vcf</code> option.

The following is an example command line to run the DRAGEN Single Cell RNA Pipeline with genotype-based demultiplexing.

```
dragen \
--enable-rna=true \
--enable-single-cell-rna=true \
--umi-source=fastq \
--scRNA-barcode-position 0_15 \
--scRNA-umi-position 16_25 \
-r reference_genomes/Mus_musculus/mm10/DRAGEN/8 \
-a reference_genomes/Mus_musculus/mm10/gtf/gencode.vM23.annotation.gtf.gz \
-l lib1_S7_L001_R2_001.fastq.gz \
--umi-fastq lib1_S7_L001_R1_001.fastq.gz \
--RGID=1 \
--RGSM=sample1 \
--output-dir=/staging/out \
--output-file-prefix=sample1 \
--scRNA-demux-detect-doublet=true \
--scRNA-demux-sample-vcf=sample.vcf
```

The following is an example command line to run the DRAGEN Single Cell RNA Pipeline with genotype-free demultiplexing.

```
dragen \
--enable-rna=true \
--enable-single-cell-rna=true \
--umi-source=fastq \
--scRNA-barcode-position 0_15 \
--scRNA-umi-position 16_25 \
-r reference_genomes/Mus_musculus/mm10/DRAGEN/8 \
-a reference_genomes/Mus_musculus/mm10/gtf/gencode.vM23.annotation.gtf.gz \
-l lib1_S7_L001_R2_001.fastq.gz \
--umi-fastq lib1_S7_L001_R1_001.fastq.gz \
```

```
--RGID=1 \
--RGSM=sample1 \
--output-dir=/staging/out \
--output-file-prefix=sample1 \
--scRNA-demux-detect-doublet=true \
--scRNA-demux-reference-vcf=sample.vcf \
--scRNA-demux-number-samples=4
```

Outputs

You can find information related to the output of genotype-based scRNA sample demultiplexing in the following three files.

The `<prefix>.scRNA.barcodeSummary.tsv` contains per-cell metrics, including cell barcodes. The following columns contain information on demultiplexing per-cell. See [Outputs on page 589](#) for more information on `<prefix>.scRNA.barcodeSummary.tsv` metrics.

Column	Description
SampleIdentity	<p>The <code>SampleIdentity</code> column can contain the following values:</p> <ul style="list-style-type: none"> • <code>sampleX</code>—The particular cell (barcode) is uniquely assigned to a sample. • <code>AMB (sampleX, sampleY)</code>—The algorithm cannot determine the sample to assign the barcode to. • <code>MIX (mixing_coef * sampleX + (100 - mixing_coef) * sampleY)</code>—The cell barcode is classified as doublet. For example, <code>MIX (50 * sampleX + 50 * sampleY)</code>.
IdentityQscore	<p>The <code>IdentityQscore</code> column contains the value used to estimate the confidence of the sample identity call. After DRAGEN determines the doublet status of the cell as <code>singlet</code>, <code>ambiguous</code>, or <code>doublet</code>, the identity Q-score is defined as $-10 * \log_{10}(\text{Probability that the assigned identity is correct, given the second most likely identity and the doublet status})$. The higher values of identity Q-score correspond to more confident sample identity calls.</p>

The `<prefix>.scRNA.demux.tsv` file contains sample demultiplexing statistics that were used to infer sample identity of each cell.

Column	Description
Barcode	The cell barcode associated with the cell.
DemuxSNPCount	The number of SNPs that the reads of the cell barcode intersect.
DemuxReadCount	The number of UMIs of the cell barcode that intersect at least one SNP.
Pure Samples	Samples from the VCF file.
BestMixtureIdentity	Mixture sample with the highest log likelihood. Only available if --single-cell-demux-detect-doublets=true.
BestMixtureLogLikelihood	The log likelihood of the best mixture sample. Only available if --single-cell-demux-detect-doublets=true.

The `<prefix>.scRNA.metrics.demuxSamples.csv` file contains per-cell metrics, similar to the metrics reported for the overall data set in `<prefix>.scRNA.metrics.csv`.

Column	Description
Passing cells	The number of cell barcodes that passed.
Fraction genic reads in cells	Counted reads assigned to the cells that passed.
Median reads per cell	Total counted reads per cell that passed the filters.
Median UMIs per cell	Total counted UMIs per cell that passed the filters.
Median genes per cell	Genes with at least one UMI per cell that passed the filters.

Cell-Hashing

DRAGEN implements several strategies for demultiplexing of data sets that represent mixtures of cells from different individuals, such as cells from different individuals pooled in one library prep or microfluidic run. One of these methods is a sample oligo-tag based method, referred to as cell-hashing.

To use cell-hashing, you must provide a cell-hashing CSV or FASTA reference file. In CSV format, the feature barcode reference file uses the following header:

`id, name, read, position, sequence, feature_type`.

- `id`—Identifier of the feature. For example, ADT_A1018.
- `name`—Name of the feature. For example, ADT_Hu.HLA.DR.DP.DQ_A1018.
- `read`—Read 1 (R1) or Read 2 (R2).
- `position`—Position on the specified read, including starting position and the length of the feature barcode. For example, a position of `0_15` represents a feature barcode that starts at position 0 and has a length of 15.
- `sequence`—DNA sequence of the feature barcode. For example, CAGCCCGATTAAGGT.
- `feature_type`—Type of the feature. For example, Antibody Capture.

Command-Line Options

To enable cell-hashing sample demultiplexing, specify the following command line options.

- `--scRNA-cell-hashing-reference`—Specify a CSV or FASTA cell-hashing reference file that contains sample-specific oligo-tags.
- `--scRNA-demux-detect-doublets`—Enable doublet detection in cell-hashing sample demultiplexing. The default value is `false`.
- `--scRNA-demux-sample-fastq`—Output sample-specific FASTQ files. See [Sample-Specific FASTQ Output Files on page 598](#) for more information.

Outputs

The `<prefix>.scRNA.barcodeSummary.tsv` file contains per-cell metrics, including cell barcodes. The following column in the `<prefix>.scRNA.barcodeSummary.tsv` contains cell-hashing per-cell information. For more information on the `<prefix>.scRNA.barcodeSummary.tsv` file, see [Outputs on page 589](#).

Column	Description
SampleIdentity	<p>The <code>SampleIdentity</code> column can contain the following values:</p> <ul style="list-style-type: none"> • <code>sampleX</code>—The particular cell (barcode) is uniquely assigned to a sample. • <code>AMB (sampleX, sampleY)</code>—The algorithm cannot determine the sample to assign the barcode to. • <code>MIX (mixing_coef * sampleX + (100 - mixing_coef) * sampleY)</code>—The cell barcode is classified as doublet. For example, <code>MIX (50 * sampleX + 50 * sampleY)</code>.

The `<prefix>.scRNA.demux.tsv` file contains sample demultiplexing statistics that were used to infer sample identity of each cell.

Column	Description
Barcode	The cell barcode associated with the cell.
Pure samples	Cell-hashing read count for each sample.

Sample-Specific FASTQ Output Files

If you have enabled either of the sample demultiplexing algorithms, you can output sample-specific FASTQ files after the sample identities for each cell is available. Use the following command line.

`--scRNA-demux-sample-fastq`

If `gzip` is specified, then the sample-specific output FASTQ files are compressed in gzip format. If `fastq` is specified, then the sample-specific FASTQ files are not compressed. The default option is none, which indicates that no sample-specific FASTQ files are output.

Feature Counting

Feature counting is a technique to profile the expression of proteins (eg, cell surface proteins or antibodies). Feature reads differ from RNA reads because the read R2 is an oligo tag corresponding to a particular type of protein, and not a transcriptomic sequence.

To enable feature counting, specify the following command line options:

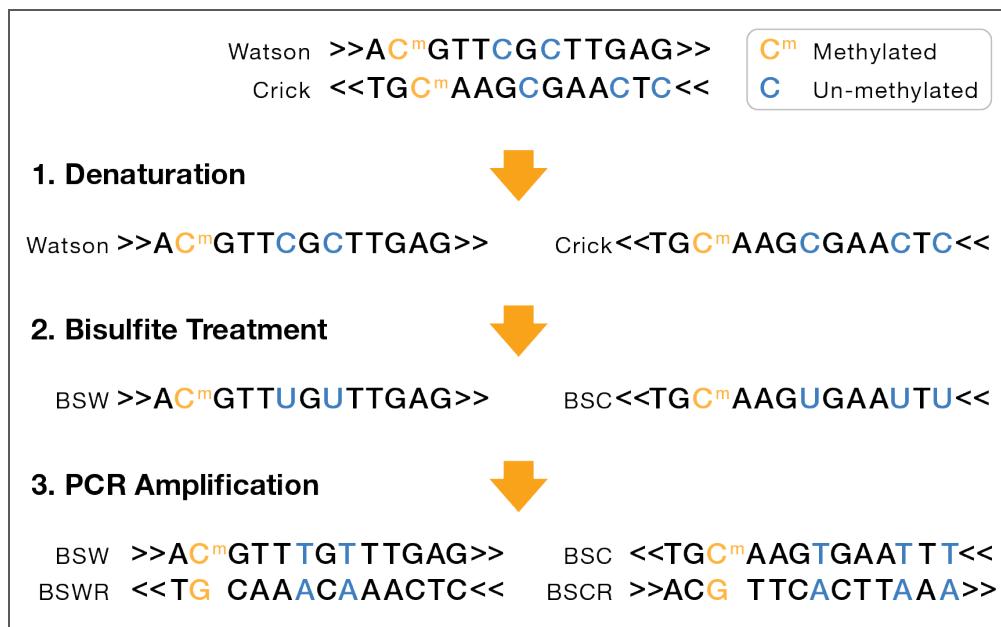
- `--scrna-feature-barcode-reference`—Specify a CSV or FASTA feature reference file that contains feature barcodes.
- `--scrna-feature-barcode-groups`—If feature reads are specified as separate FASTQ files, specify a comma-separated list of read groups that correspond to feature FASTQ files.

Output of feature counting is appended to the output expression matrix. The extended expression matrix contains additional rows corresponding to features.

DRAGEN Methylation Pipeline

The epigenetic methylation of cytosine bases in DNA can have a dramatic effect on gene expression, and bisulfite sequencing is the most common method for detecting epigenetic methylation patterns at single-base resolution. This technique involves chemically treating DNA with sodium bisulfite, which converts unmethylated cytosine bases to uracil, but does not alter methylated cytosines. Subsequent PCR amplification converts any uracils to thymines.

A bisulfite sequencing library can either be nondirectional or directional. For nondirectional, each double-stranded DNA fragment yields four distinct strands for sequencing, post-amplification, as shown in the following figure:



- Bisulfite Watson (BSW), reverse complement of BSW (BSWR),
- Bisulfite Crick (BSC), reverse complement of BSC (BSCR)

For directional libraries, the four strand types are generated, but adapters are attached to the DNA fragments such that only the BSW and BSC strands are sequenced (Lister protocol). Less commonly, the BSWR and BSCR strands are selected for sequencing (eg, PBAT).

BSW and BSC strands:

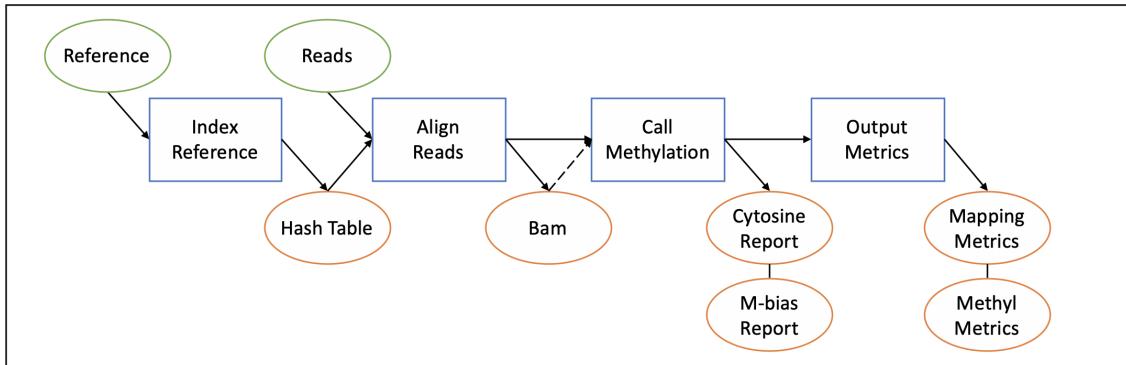
- A, G, T: unchanged
- Methylated C remains C
- Unmethylated C converted to T

BSWR and BSCR strands:

- Bases complementary to original Watson/Crick A, G, T bases remain unchanged.

- G complementary to original Watson/Crick methylated C remains G.
- G complementary to original Watson/Crick unmethylated C becomes A.

Therefore, several steps are performed to map methylation, as shown in the following flowchart:



Details on each part of the mapping process can be found in this section.

Mapping Method Options

DRAGEN supports two methods for methylation mapping, single-pass and multi-pass.

Single-pass Methylation Mapping

Single-pass methylation mapping is the default and recommended mapping method. In single-pass, DRAGEN performs only one alignment run. During alignment, the mapper considers all possible base and reference conversions for the read and emits the single best alignment to a particular methylation strand if one exists. Any read (pair) that did not have a single best scoring alignment across all methylation strands tested appears in the output BAM with `MAPQ = 0`. The output BAM in single-pass might contain mapped reads that do not have the XM, XR, and XG methylation tags. Methylation data from reads that do not have methylation tags are not tallied into the reports or metrics files.

`--enable-methylation-calling` must be set to true to enable single-pass methylation mapping.

A read must meet the following requirements to have methylation tags in the output BAM and to get tallied into the reports and metrics:

- The read and its mate (if applicable) are mapped with MAPQ above the value specified using `--methylation-mapq-threshold`. The default value is 0.
- The read is not part of an improper pair.

Multi-pass Methylation Mapping (Deprecated)

In multipass methylation mapping, multiple alignment runs are executed sequentially. Alignments from each run are stored on a disk. After all alignment runs are complete, the stored results are compared and the best alignments are chosen. DRAGEN excludes any read (pair) that did not have a single best

scoring alignment across all methylation strands tested from the output BAM. Any reads that appear in the output BAM have methylation BAM tags. If you enabled report generation during mapping, the reads are included in the methylation reports.

To enable multipass methylation mapping, set `methylation-mapping-implementation=multi-pass`.

When `--enable-methylation-calling` is set to true, DRAGEN analyzes the multiple alignments to produce a single methylation-tagged BAM file. When `--enable-methylation-calling` is set to false, DRAGEN outputs a separate BAM file per alignment run.

If `--methylation-match-bismark=true` is set, then the position of paired-end reads that map to CTOT or CTOB strands are flipped within their template. The read that appears as Read 1 in the output BAM will be from the second FASTQ file.

Build a Methylation Hash Table

To run DRAGEN methylation pipeline, you need to build a methylation-specific hash table. Both multi-pass and single-pass options require methylation hash tables. The type of hash table must match the method used during alignment (`methylation-mapping-implementation: single-pass | multi-pass`). For more information on multi-pass and single-pass methods, see [Mapping Method Options on page 601](#).

If using the single-pass method, make sure that your hash table meets the following requirements:

- Set `methylation-mapping-implementation: single-pass`.
- Set `ht-methylated-combined=true` to generate a combined hash table in the `methyl_CONVERTED` subdirectory.
- Single-pass mode supports alt-contig references. Each contig in the original FASTA appears twice in the combined reference genome files, one time with each conversion type.

If using the multi-pass method, make sure that your hash table meets the following requirements:

- Set `methylation-mapping-implementation: multi-pass`.
- Set `ht-methylated=true` to generate the `CT_CONVERTED` and `GA_CONVERTED` subdirectories.
- Multi-pass mode does not support alt-contig references. The subdirectories contain the genome index for C to T and G to A conversion.

You can set `ht-methylated-combined=true` and `ht-methylated=true` together to create a hash table directory that can be used for both methylation-mapping methods.

The following is an example command line for a single-pass hash table.

```
dragen --build-hash-table true \
--output-directory=sample.output.directory \
--ht-reference=sample.input.fa \
--ht-num-threads 40 \
```

```
--ht-methylated=true \
--ht-methylated-combined=true \
--ht-seed-len 27
```

The `--ht-seed-len 27` option is important for optimal results. The `--ht-num-threads` option enables multiple numbers of threads for faster building.

DRAGEN Methylation Calling

Different methylation protocols require the generation of two or four alignments per input read, followed by an analysis to choose a best alignment and determine which cytosines are methylated. DRAGEN can automate this process by generating a single output BAM file with Bismark-compatible tags (XR, XG, and XM) that can be used for methylation calling and other downstream workflows.

When the `--methylation-protocol` option is set to a valid value other than none, DRAGEN automatically produces the required set of alignment runs. Each alignment run includes the appropriate base conversions on the reads, base conversions on the reference, and constraints on whether reads must be forward-aligned or reverse complement (RC) aligned with the reference.

The following options are automatically configured.

- `--generate-md-tags true`
- `--Aligner.global 1`
- `--Aligner.no-unpaired 1`
- `--Aligner.aln-min-score 0`
- `--Aligner.min-score-coeff -0.2`
- `--Aligner.match-score 0`
- `--Aligner.mismatch-pen 4`
- `--Aligner.gap-open-pen 6`
- `--Aligner.gap-ext-pen 1`
- `--Aligner.supp-aligns 0`
- `--Aligner.sec-aligns 0`
- `--preserve-map-align-order true` (multipass only)
- `--seed-density 1` (single-pass only)

Because global alignments (end-to-end in the reads) are generated, DRAGEN recommends trimming any artifacts introduced by library prep and adapter sequences.

The following table describes these alignment runs:

Protocol	BAM	Reference	Read 1	Read 2	Orientation Constraint
directional					
1	C->T		C->T	G->A	Forward-only
2	G->A		C->T	G->A	RC-only
nondirectional, or directional-complement					
1	C->T		C->T	G->A	Forward-only
2	G->A		C->T	G->A	RC-only
3	C->T		G->A	C->T	RC-only
4	G->A		G->A	C->T	Forward-only
PBAT					
3	C->T		G->A	C->T	RC-only
4	G->A		G->A	C->T	Forward-only

In directional protocols, the library is prepared such that only the BSW and BSC strands are sequenced. Thus, alignment runs are performed with the two combinations of base conversions and orientation constraints best suited for these strands (directional runs 1 and 2 above).

With nondirectional protocols, reads from each of the four strands are equally likely, so alignment runs must be performed with two more combinations of base conversions and orientation constraints (nondirectional runs 3 and 4 above).

In PBAT protocols, the library is prepared so only the BSWR and BSCR strands are sequenced. Only two alignment runs are performed with the combinations of base conversions and orientation constraints best suited for these strands (runs 3 and 4).

The directional-complement protocol can also be used for PBAT or similar libraries where mainly the BSWR and BSCR strands are sequenced. With this protocol, all four aligner runs are performed, but relatively few good alignments are expected from the runs for the BSW and BSC strands, so DRAGEN is automatically tuned to a faster analysis mode for those runs.

The following is an example DRAGEN command line for the directional protocol:

```
dragen --enable-methylation-calling true \
--methylation-protocol directional \
--ref-dir /staging/ref/mm10/methylation --RGID RG1 --RGCN CN1 \
--RGLB LIB1 --RGPL illumina --RGPU 1 --RGSM Samp1 \
--intermediate-results-dir /staging/tmp \
-1 /staging/reads/samp1_1.fastq.gz \
-2 /staging/reads/samp1_2.fastq.gz \
--output-directory /staging/outdir \
--output-file-prefix samp1_directional_prot
```

Using Bismark for Methylation Calling

The recommended approach to methylation calling is to automate the multiple required alignment runs and add the XM, XR, XG tags. If using the multipass mapping method, you can generate a separate BAM file for each of the constraints and conversions needed by the methylation protocol by setting `--enable-methylation-calling` to false. The BAM files can be used as input into Bismark for methylation calling. Contact Illumina Technical Support for more information.

In this mode, a single DRAGEN run produces multiple BAM files in the directory specified by `--output-directory`. Each BAM file contains alignments in the same order as the input reads. You cannot enable sorting or duplicate marking for these runs. Alignments include MD tags and have /1 or /2 appended to the names of paired-end reads for Bismark-compatibility. The BAM files use the following naming conventions:

- Single-end reads—`output-directory/output-file-prefix.{CT,GA}read{CT,GA}reference.bam`
- Paired-end-reads—`output-directory/output-file-prefix.{CT,GA}read1{CT,GA}read2{CT,GA}reference.bam,`

Where `output-directory` and `output-file-prefix` are specified by the corresponding options, and CT and GA correspond to the base conversions listed in the table above.

Bismark does not have a directional-complement mode, but you can process such samples using Bismark's nondirectional mode with the expectation that runs 1 and 2 produce very few high scoring alignments.

Sort and Duplicate Reads Options

DRAGEN supports sorting and duplicate marking/removal for methylated reads during the alignment phase. To enable sort and duplicate read options, first check the `--methylation-mapping-implementation`. The sort and duplicate options behave differently depending on the pass mode of choice.

For the default single-pass mode, and end-to-end (ie., FASTQ->BAM->cytosine report) run can be performed as follows:

- To generate sorted alignment output (in BAM format), set `--enable-sort` to true.
- To detect duplicate reads, set `--enable-duplicate-marking` to true.
- [Optional]To remove duplicate reads, set `--remove-duplicates` to true.
- [Optional]Set `--methylation-generate-cytosine-report` and `--methylation-generate-mbias-report` to either false or true according to user need.

For the legacy multi-pass mode, perform two separate runs as follows:

1. Set the options for the first run as follows.
 - To generate sorted alignment output (in BAM format), set `--enable-sort` to true.
 - To detect duplicate reads, set `--enable-duplicate-marking` to true.

- [Optional] To remove duplicate reads, set `--remove-duplicates` to true.
- Set `--methylation-generate-cytosine-report` and `--methylation-generate-mbias-report` to false.

These options behave the same as in DNA alignment. For example, if `--enable-duplicate-marking` is set to true, `--enable-sort` is true.

2. Set the options for the second run as follows.

- Use the sort/markdup/dedup alignment output from the previous run for `-b/--bam-input`.
- Set `--methylation-reports-only` to true.
- Set `--enable-sort` to false.
- To generate cytosine report, set `--methylation-generate-cytosine-report` to true.
- To generate M-bias report, set `--methylation-generate-mbias-report` to true.

During the second step, methylated bases from reads that have XM, XR, and XG tags are tallied into reports. Reads that do not have methylation tags are ignored. If the sort and duplicate reads options are set to false, only one end-to-end run is necessary to generate an alignment file, cytosine report, and M-bias report.

By default, DRAGEN methylation performs strand-aware dedup in concordance with Bismark. Strand-aware dedup partitions the mapped reads into four groups, one per methylation strand. Within each group, DRAGEN performs a normal dedup. For paired reads, the strand of the pair is defined as the strand of the first read in the pair.

The following example demonstrates strand-aware dedup for paired-end reads. The example pairs all map to the same position, but the first read in each pair (BAM flag 83 and 99) is mapped to a different methylation strand, as shown by the different values of the XR and XG tags. None of these pairs are marked as duplicates.

```
pair1 83 lambda 44001 60 150M = 43651 ... XR:Z:CT XG:Z:GA
pair1 163 lambda 43651 60 150M = 44001 ... XR:Z:GA XG:Z:GA
pair2 83 lambda 44001 60 150M = 43651 ... XR:Z:GA XG:Z:CT
pair2 163 lambda 43651 60 150M = 44001 ... XR:Z:CT XG:Z:CT
pair3 147 lambda 44001 60 150M = 43651 ... XR:Z:GA XG:Z:CT
pair3 99 lambda 43651 60 150M = 44001 ... XR:Z:CT XG:Z:CT
```

Unique Molecular Identifiers (UMI) Support

DRAGEN support FASTQ files that contain UMI barcodes during the alignment phase. The principle and requirement are identical to DNA UMI. During library prep, (methyl-treated) DNA fragments could be barcoded by unique molecular identifiers, so that true signals from the original fragments can be separated from PCR error and sequencing error, which enables more accurate methylation calling. The *.fastq files need to have UMI barcode in 7th field of the QNAME. See the following example.

```
@NS500561:434:H5LC2BGXJ:1:11101:10798:1359:CACATGA+ACATTC
1:N:0:TGGTACCTAA+AGTACTCATG
```

To enable UMI, either set `--umi-enable` true if you are using random UMI (common), or set `--tso500-solid-umi` true if you are using the same non-random UMIs as the TSO500 solid panel. If so, read collapsing will be performed among reads with the same UMI that are mapped to the same genomic location at the same strand, either from top (OT/CTOT) or bottom (OB/CTOB) strand.

i | UMI is not supported in the deprecated multi-pass mode, so make sure `--methylation-mapping-implementation` single-pass is set.

For more information, refer to [Unique Molecular Identifiers on page 377](#).

Using TAPS Support

TET-Assisted Pyridine Borane Sequencing (TAPS) is a new assay which directly converts methylated C to T, whereas typical bisulfite conversion converts unmethylated C to T. This approach preserves genomic complexity and uses less destructive chemicals to enable lower input DNA.

To enable analysis of FASTQ data generated through TAPS, set `--methylation-TAPS` to true. By default, the option is false. This option is performed only during the alignment step and is not necessary when generating methylation cytosine and M-Bias reports from an existing BAM.

Methylation-Related BAM Tags

When `--enable-methylation-calling` is set to true, DRAGEN analyzes the alignments produced for the configured `--methylation-protocol` and generates a single output BAM file that includes methylation-related tags for all mapped reads. As in Bismark, reads without a unique best alignment are excluded from the output BAM. The added tags are as follows.

Tag	Brief Description	Description
XR:Z	Read conversion	For the best alignment, which base-conversion was performed on the read: CT or GA.
XG:Z	Reference conversion	For the best alignment, which base-conversion was performed on the reference: CT or GA
XM:Z	Methylation call	A byte-per-base methylation string.

The XM:Z (methylation call) tag contains a byte that corresponds to each base in the sequence of the read. Each position that does not involve a cytosine contains a period (.). Each position that does

involve a cytosine contains a letter. The letter indicates the context (CpG, CHG, CHH, or unknown). The case indicates methylation. Methylated positions use upper-case and unmethylated positions use lower-case. The letters used at cytosine positions are as follows.

Character	Methylated?	Context
.	Not cytosine	Not cytosine
z	No	CpG
Z	Yes	CpG
X	No	CHG
X	Yes	CHG
h	No	CHH
H	Yes	CHH
u	No	Unknown
U	Yes	Unknown

Methylation Cytosine and M-Bias Reports

You can use DRAGEN to generate a genome-wide cytosine methylation report. Your command line options settings depend on if you are running using FASTQ through the aligner or a prealigned BAM that already contains the methylation tags.

- For FASTQ input, set `--methylation-generate-cytosine-report=true`
- For BAM input, set `--methylation-reports-only=true`

To keep all cytosines from your reference in the `CX_report`, even if they are not included in the input sequences, set `--methylation-keep-ref-cytosine true`. The default value is false. Setting this option to true increases run time and the `CX_report` file size.

To compress the cytosine report, set `--methylation-compress-cx-report = true`. The default value is false. DRAGEN outputs a compressed `*.CX_report.txt.gz`, instead of a `*.CX_report.txt`.

The position and strand of each C in genome are given in the first three fields of the report. A record with a - in the strand field is used for a G in the reference FASTA. The counts of methylated and unmethylated Cs covering the positions are given in the fourth and fifth fields. The C context in the reference (CG, CHG, or CHH) is given in the sixth field. The trinucleotide sequence context is given in the last field (eg, CCC, CGT, CGA, and so on) The cytosine report only includes records for positions that have one or more spanning alignments. The following is an example cytosine report record:

```
chr2    24442367    +    18    0    CG    CGC
```

To generate an M-bias report, set `--methylation-generate-mbias-report` to true. This report contains three tables for single-ended data with one table for each C-context and six tables for paired-end data. Each table is a series of records, with one record per read base position. For example, the first record for the CHG table contains the counts of methylated Cs (field 2) and unmethylated Cs (field 3)

that occur in the first read base position, and restricts to those reads in which the first base is aligned to a CHG location in the genome. Each record of a table also includes the percent methylated C bases (field 4) and the sum of methylated and unmethylated C counts (field 5).

The following is an example M-bias record for read base position 10:

10	7335	2356	75.69	9691
----	------	------	-------	------

For data sets with paired-end reads that overlap, both the cytosine and M-bias reports do not report any Cs in the second read that overlaps the first read. In addition, 1-based coordinates are used for positions in both reports.

If using the multipass mapping method or reports-only mode, you can match the `bismark_methylation_extractor` cytosine and M-bias reports generated by Bismark version 0.19.0, by setting the `--methylation-match-bismark` option to true. The ordering of records in Bismark and DRAGEN cytosine reports might differ. DRAGEN reports are sorted by genomic position.

Output Metrics

The quality of each methylation run can be summarized in the following two metric files.

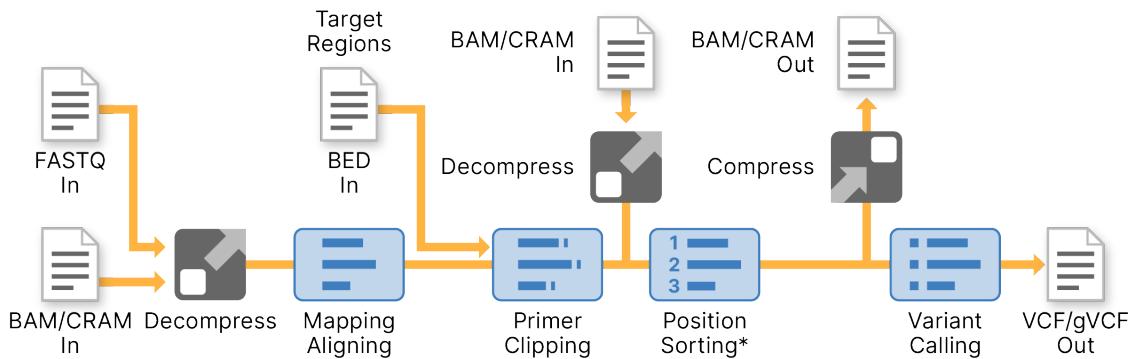
- `*.mapping_metrics.csv`—Contains mapping-specific metrics that are generated for the alignment phase, including benchmarks like number of total reads, aligned reads, deduped reads, base quality, etc.
- `*.methyl_metrics.csv`—Contains methylation-specific metrics that are generated for the methylation calling phase, including benchmarks like the total number of cytosines analyzed, count and rate of methylation in each cytosine context, strand of the best alignment, etc.

DRAGEN Amplicon Pipeline

Amplicon sequencing is a highly targeted approach that enables you to analyze genetic variation in specific genomic regions. The ultradeep sequencing of PCR products (amplicons) allows you to efficiently identify and characterize variants. This method uses oligonucleotide probes designed to target and capture regions of interest, followed by next-generation sequencing (NGS).

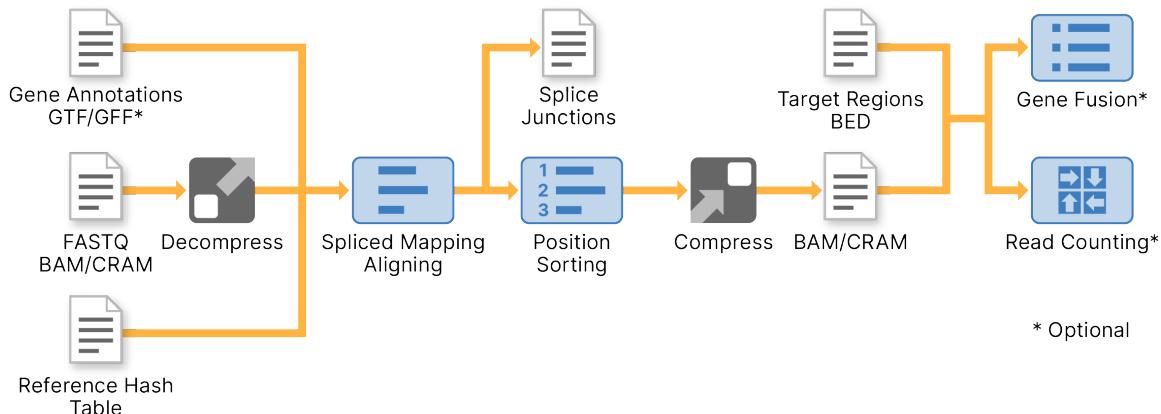
The Amplicon Pipeline supports both DNA and RNA data. The Amplicon Pipeline turns off duplicate marking because there are only a few unique start and end postions for fragments from an amplicon target due to the assay.

The DNA Amplicon Pipeline uses the DRAGEN DNA Pipeline by including an additional step after mapping and aligning to soft-clip primers and rewrite alignments. If the target amplicon is found, DRAGEN tags each alignment with the target amplicon and performs soft-clipping the primer sequences. DRAGEN performs tagging by adding an `XN:Z:<amplicon name>` tag to the output BAM/CRAM record. Soft-clipping makes sure that the primer sequences do not contribute to the variant calls.



* Optional

The RNA Amplicon Pipeline uses the DRAGEN RNA Pipeline. Amplicon-specific parameters are set for fusion calling, including a fusion scoring model trained on RNA amplicon data. Small variant calling is not supported in RNA amplicon mode.



* Optional

Amplicon BED File

The DRAGEN Amplicon Pipeline requires an amplicon BED file and all input files required by the DRAGEN DNA pipeline. Each row in an amplicon BED file describes an amplicon target. The following fields are required.

Field	Description
chrom	The name of the chromosome.
chromStart	The 0-based inclusive start position of the target, excluding the primer.
chromEnd	The 0-based exclusive end position of the target, excluding the primer.
name	The name of the amplicon target.
gene	[Optional] The gene ID.
targetType	[Optional] The target type.

The default segmentation of DNA amplicon mode in copy number variant calling is `bed`, and can be modified using `--cnv-segmentation-mode`. The CNV segmentation bed is gene-level and auto-generated based on the gene ID column in the amplicon BED file. In RNA amplicon mode, `targetType` is used to identify fusion targets where `targetType` is `Fusion`. The gene IDs for fusion targets are collected and written to an output file. The default value of `--rna-gf-enriched-genes` is then set to this file containing fusion gene IDs. A candidate fusion is required to have both partner genes in the gene list. Base-level and read-level coverage is calculated for each region in the amplicon BED file. It is recommended that the fusion targets are commented to avoid competition with gene expression targets.

DRAGEN DNA Amplicon Settings

To use the DNA amplicon pipeline, set `--enable-dna-amplicon` to `true`. Use `--amplicon-target-bed` to specify the path to your amplicon BED file.

To enable small variant calling, set `--enable-variant-calling` to `true`.

To enable copy number variant calling, set `--enable-cnv` to `true`. GC bias correction when generating target counts is enabled by default. The generation of the target counts for the normal samples should also have identical command line options with the case sample under analysis.

To enable structural variant calling, set `--enable-sv` to `true`.

The target small variant calling BED input is set to amplicon BED file by default and can be modified using `--vc-target-bed`. The CNV segmentation bed is auto generated based on the gene ID column in the amplicon BED file and can be modified using `cnv-segmentation-bed`. See [CNV Targeted Segmentation](#) for more information.

The amplicon pipeline can be run in either germline or somatic mode. Specify a tumor-only or tumor-normal input for the somatic mode. For more information see [Somatic Mode on page 126](#) and [Somatic Mode Options on page 127](#). For more information on the multicaller (germline & somatic) workflows, see [Multicaller Workflows](#). If calling somatic small variants, set `--vc-use-somatic-hotspots` to `false`.

By default the maximum amplicon primer length is set to 50. You can specify a different value using `--amplicon-primer-length`. The parameter affects whether an alignment is assigned to an amplicon target. If an alignment starts inside the primer region of the amplicon target, the alignment is assigned to the amplicon. For a properly paired alignment, both the alignment and the mate must come from the same amplicon target.

```
|-- primer --|-- amplicon target --|-- primer --|
----- read ----->
          <----- read ----->
```

The following is an example command line to run the DRAGEN DNA Amplicon Pipeline with copy number, structural variant, and germline small variant calling.

```
dragen
--enable-dna-amplicon true
--enable-map-align=true
--enable-sort=true
--enable-map-align-output=true -r reference_genomes/Hsapiens/hg19_alt_
aware/DRAGEN/8
--amplicon-target-bed=CancerHotSpot-v2.dna_manifest.20180509.bed
--enable-variant-caller=true
--enable-cnv=true
--enable-sv=true
--fastq-file1=read1.fastq.gz
--fastq-file2=read2.fastq.gz
--RGSM NA12878 --RGID 1
--output-directory=/staging/out
--output-file-prefix=NA12878
```

DRAGEN RNA Amplicon Settings

To use the RNA amplicon pipeline, set `--enable-rna-amplicon` to `true`. Use `--amplicon-target-bed` to specify the path to your amplicon BED file.

We do not recommend enabling RNA quantification to produce the `.sf` quantification output files as a panel-specific GTF file is usually not used. The `.target_bed_read_cov_report.bed` read-level coverage output file should be used instead. This file is automatically produced when map/align is output enabled.

To enable RNA gene fusion calling, set `--enable-rna-gene-fusion` to `true`. Fusion calling parameters are automatically set in RNA amplicon mode but can be overridden in the command line. If fusion targets are not listed in the amplicon BED file, users can explicitly set `--rna-gf-enriched-genes` to a file containing fusion gene IDs or symbols.

The following is an example command line to run the DRAGEN Compute Server RNA Amplicon Pipeline with gene fusion calling.

```
dragen --enable-rna-amplicon true --enable-map-align=true --enable-sort=true --
enable-map-align-output=true -r reference_genomes/Hsapiens/hg19_alt_
aware/DRAGEN/8 --amplicon-target-bed=Myeloid.rna_manifest.20201014.bed --
enable-rna-gene-fusion=true --ann-sj-file=gencode.v19.annotation.gtf --output-
format=BAM --fastq-file1=read1.fastq.gz --fastq-file2=read2.fastq.gz --RGSM
Seraseq --RGID 1 --output-directory=/staging/out --output-file-prefix=Seraseq
```

Tools and Utilities

BCL Data Conversion

The Illumina BCL Convert is a standalone local software application that converts the Binary Base Call (BCL) files produced by Illumina sequencing systems to FASTQ files. The DRAGEN product includes hardware accelerated BCL conversion on the DRAGEN platform, which results in improved run times compared to BCL Convert pure software execution.

The DRAGEN BCL conversion is designed to output FASTQ files that match `bcl2fastq2` v2.20 output. DRAGEN supports direct conversion from `.BCL` to the compressed `FASTQ.ORA` format in order to reduce the `FASTQ.GZ` file size by a ratio up to 5. Refer to [DRAGEN ORA compression from BCL](#) for proper usage.

DRAGEN Analysis app supports the following features.

- Demultiplexing samples by barcode with optional mismatch tolerance
- Supports adapter sequence masking or trimming with adjustable matching stringency.
- Supports UMI sequence tagging and optional trimming
- Optional Output of FASTQ files for index reads (in gzipped or FASTQ.ORA files)
- Optional Combines all lanes to the same FASTQ output files
- Supports high sample count (100,000)
- Supports UMI sequences in index reads
- Eliminates skew as the result of adapter sequence trimming by using the `MinimumAdapterOverlap` setting
- Supports combined (default, compatible with `bc2fastq2`) or independent (strict) enforcement of demux conflict detection
- Supports mixed pools by specifying settings for each sample (`OverrideCycles`, `Adapters`, etc)
 - Converts all data in a single invocation
 - Automatically detects barcode conflicts, including between pools
 - Allows single and dual-index kits to be mixed
 - Undetermined files correctly contain reads that do not map to any sample in any pool
- Outputs metrics for demultiplexing, quality scores, adapter trimming, unmapped barcodes, and index-hopping detection.
- Outputs per-cycle adapter metrics and per-tile quality & demultiplex metics
- Converts a subset of tiles specified by regular-expressions using an allow list, a block list, or both.
- Better support for legacy applications based upon `bcl2fastq2` (all off by default):

- Output metrics in bcl2fastq2 Stats directory format in addition to csv
- Support FindAdaptersWithIndels setting to match bcl2fastq2 default output
- Support fastq subdirectories named by sample project, sampleID, & sampleName

System Requirements

When not running on the DRAGEN platform, the following requirements should be noted:

- Minimum 64 GB of RAM (less RAM is required for some smaller flow cell input types, such as NextSeq and iSeq)
- Storage requirements: sufficient storage for BCL input and FASTQ output on each source and destination storage device (no intermediate output is generated during BCL conversion)
- Linux CentOS 6 or higher
- Root access

Installation

For DRAGEN products, BCL conversion functionality is included. When using the separate BCL Convert application, note the following:

BCL Convert is installed from an RPM package downloaded from the Illumina support site. Install the RPM package using one of the following commands:

- To install the software in the default location, enter: `rpm --install <rpm package-name>`
- To specify a custom install location, enter: `rpm --install --prefix <user-specified directory> <rpm package-name>`

The default installation places the executable at `/usr/local/bin/bcl-convert`.

Run Requirements

BCL Convert and DRAGEN require the following files to be present in the run folder to perform BCL conversion:

- BCL files (*.bcl, *.cbcl)
- Filter files (*.filter)
- Position files (*.locs, *.clocs, or s.locs)
- Aggregated files (*.bci) as applicable
- The RunInfo.xml file
- The config.xml file (for older systems) if applicable

- The SampleSheet.csv file -- Supports v1 and v2. See the Sample Sheet section below for more details.

Command Line Options

The following command contains the required BCL conversion options for DRAGEN.

```
dragen
--bcl-conversion-only true
--bcl-input-directory <...>
--output-directory <...>
```

The following commands are required for bcl-convert:

```
bcl-convert
--bcl-input-directory <...>
--output-directory <...>
```

There are many optional command line arguments as well. The following is a list of all command line options:

Software Behavior

Option	Description	Default
--bcl-conversion-only-true	(DRAGEN only) Required for BCL conversion to FASTQ files in the DRAGEN executable	Not applicable
--bcl-input-directory	A main command line option that indicates the path to the run folder directory.	Not applicable
--output-directory	A required command line option that indicates the path to demultiplexed FASTQ output. The directory must not exist, unless -f --force is specified.	Not applicable
--sample-sheet	[Optional] If different from the default, the option indicates the path to the sample sheet to specify the sample sheet location and name.	<--bcl-input-directory>/SampleSheet.csv
--run-info	Overrides the path to the RunInfo.xml file.	--bcl-input-directory

Option	Description	Default
--strict-mode	[Optional] If true, the option aborts the program if any filter, LOCSlocs, BCLbcl, or BCLbc lane files are missing or corrupt. If false, the option continues processing if any filter, LOCSlocs, BCLbcl, or BCLbc lane files are missing. Returns a warning message for each missing or corrupt file.	false
--first-tile-only	If true, the option only converts the first tile of input (for testing and debugging).	false
--bcl-only-lane <#>	Convert only the specified lane in this conversion run. Default convert all lanes.	
-f --force	Convert to output directory even if the directory exists (force).	
--bcl-use-hw-false	Allows concurrent execution of BCL conversion with DRAGEN analysis. Do not use DRAGEN FPGA acceleration during BCL conversion.	Not applicable
--bcl-sampleproject-subdirectories	[Optional] If true, the option allows creation of Sample_Project subdirectories as specified in the sample sheet. To use the Sample_Project column in the data section, this option must be set to true for the Sample_Project column in the data section to be used.	false
--no-lane-splitting-true	Output all lanes of a flow cell to the same FASTQ files consecutively.	false
--no-lane-splitting	Consolidates FASTQ files across lanes. Each sample is provided into the same file on a per-read basis. <ul style="list-style-type: none"> • Must be true or false. • Only allowed when Lane column is excluded from the sample sheet. 	false

Option	Description	Default
--bcl-only-matched-reads--true	Disable outputting unmapped reads to FASTQ files marked as Undetermined.	false
--tiles	Only converts tiles that match a set of regular expressions.	Not applicable
--exclude-tiles	Do not convert tiles that match a set of regular expressions, even if included in --tiles.	Not applicable
--no-sample-sheet true	Operate without a sample sheet (no demultiplexing or adapter trimming supported).	False
	This option is not supported for conversion to FASTQ.ORA.	
--output-legacy-stats true	Output metrics in bcl2fastq2 Stats directory format in addition to csv.	False
--sample-name-column-enabled true	Use Sample_Name Sample Sheet column for *.fastq file names in Sample_Project subdirectories (requires bcl-sampleproject-subdirectories true as well).	False
--fastq-gzip-compression-level [0-9]	Set gzip compression level for software-compressed FASTQ files. This setting will not impact blocks compressed by FDGA hardware.	1
-h, --help	Produces a help message and exits the application.	Not applicable
-V, --version	Produces a help message and exits the application.	Not applicable
--ora-reference	Required to output compressed FASTQ.ORA files. Specify the path to the directory that contains the compression reference and index file.	Not applicable

Option	Description	Default
--fastq-compression-format	Required for DRAGEN ORA compression to specify the type of compression: use dragen for regular DRAGEN ORA compression, or dragen-interleaved for DRAGEN ORA paired compression.	
--num-unknown-barcodes-reported	# of Top Unknown Barcodes to output (1000 by default)	
--bcl-validate-sample-sheet-only	Only validates RunInfo.xml & SampleSheet files. It only creates bcl-validate-sample-sheet-only and does not produce FASTQ files.	

Software Performance

The following additional options can be used to manually control performance. Use of these options might reduce performance or result in analysis failure, and it is recommended to use the default settings. Contact Illumina Technical Support if issues occur.

Total CPU heavy threads should be less than the number of HW cores you want to use for other processes. The following formula gives the total number of CPU heavy threads used by DRAGEN:

Option	Description	Default
--shared-thread-odirect-output true	Switch to an alternate file output method that is optimized for sample counts greater than 100,000. This option is not recommended for lower sample counts and/or if using distributed file system output targets such as GPFS or Lustre.	
--bcl-num-parallel-tiles <#>	Number of tiles processed in parallel.	Determined dynamically
--bcl-num-conversion-threads <#>	Number of conversion threads per tile.	Determined dynamically
--bcl-num-compression-threads <#>	Number of CPU threads for gzip-compressing FASTQ output.	Determined dynamically

Option	Description	Default
--bcl-num-decompression-threads <#>	Number of CPU threads for decompressing input BCL files.	Determined dynamically
--bcl-num-ora-compression-threads-per-file <#>	Optional for DRAGEN ORA compression. Set the number of threads used per file files.	Default: 10 Maximum: 24
--bcl-num-ora-compression-parallel-files <#>	Optional for DRAGEN ORA compression. Set the number of files processed in parallel.	Default: 6 Maximum: 96

It is recommended to only adjust CPU threads when reducing cores used on a shared machine. The total number of CPU-intensive threads used will be: `--bcl-num-parallel-tiles * --bcl-num-conversion-threads + --bcl-num-compression-threads + --bcl-num-decompression-threads`.

Tile Filtering

DRAGEN Analysis app uses two command line options to control which tiles are converted. The correct command line depends upon the tile list expression.

- `--tiles`—Specifies which tiles to include for analysis.
- `--exclude-tiles`—Specifies which tiles to exclude from analysis.

This feature is a replacement for `tiles`, `ExcludeTiles`, and `ExcludeTilesLaneX` in `bcl2fastq2`. Both `--tiles`, and `--exclude-tiles` use a tile name (single regular expression) format. For example:

- `--tiles 1101`—Includes the first tile on the first surface and first swath of every lane.
- `--tiles 11101`—For NextSeq 500/550 system only. Includes the first tile on the first surface and first swath of every lane.
- `s_`—Used to specify a lane. For example:
 - `--tiles s_2`—Converts all tiles of lane 2.
 - `--exclude-tiles s_2_1101`—Excludes the first tile on the first surface and first swath of lane 2.

Use square brackets and single digits separated by a hyphen to specify a range. For example:

- `[1-2]101`—Select the first tile of both sides of the flow cell.
- `s_[1-2]_[1-2][0-9][0-9][0-9]5`—Selects all tiles ending with 5 from surfaces 1–2 and all swaths for lanes 1–2.

To specify multiple tile expressions, use +.

- `s_1_1102+s_[2-8]`—Includes the second tile on the first surface and first swath of lane 1 and all tiles in lanes 2–8.

Every component of the regular expression (as separated by +) used for `tiles` must match at least one tile entry in the input `RunInfo` tile list. Every term for `exclude-tiles` must match at least one tile entry in the set produced by `tiles` if that option is also used, or the `RunInfo` tile list.

DRAGEN ORA compression from BCL

BCL files can be converted into FASTQ.ORA using two different methods, which cannot be used at the same time.

Method 1: Using command line without a sample sheet:

1. Set the path to the directory that contains the compression reference and index file with the `--ora-reference` command.
2. Specifying the type of DRAGEN ORA compression with the `-fastq-compression-format` command. The values can be either `dragen` regular DRAGEN ORA compression or `dragen-interleave` for DRAGEN ORA paired compression.

Method 2: Using command line with a sample sheet:

1. Set the path to the directory that contains the compression reference and index file with the `--ora-reference` command.
2. Specify the type of DRAGEN ORA compression in the sample sheet. See [Sample Sheet Settings](#) for proper syntax.

The reference and index files for ORA compression are available at [DRAGEN Bio-IT Platform Product Files](#).

For information about how to use FASTQ.ORA files see [Input File Types](#).

Interleaved compression

The interleaved DRAGEN ORA compression improves the compression up to 10% vs. DRAGEN ORA regular compression. Interleaved compression is enabled by setting `--fastq-compression-format` to `dragen-interleaved`. The paired-read file from the nth line of `fastq-list.csv` generated by the BCL convert tool are compressed together into a single `fastq.ora` file, with the name `<filename before "R">-interleaved<_suffix>.fastq.ora`. `<_suffix>` is optional.

If decompressing an ORA file contains paired data, the file is automatically decompressed into two separate files. To map an ORA file that contains paired interleaved data with the DRAGEN mapper, use the `--interleaved` option during map/align.

Regular Compression Example

The following example contains the required BCL conversion options to run a regular DRAGEN ORA compression from BCL:

```
dragen
--bcl-conversion-only true
--bcl-input-directory <...>
--sample-sheet <...>
--ora-reference <...>
--fastq-compression-format dragen
--output-directory <...>
```

Interleaved Compression Example

The following example contains the required BCL conversion options to run interleaved DRAGEN ORA compression from BCL:

```
dragen
--bcl-conversion-only true
--bcl-input-directory <...>
--sample-sheet <...>
--ora-reference <...>
--fastq-compression-format dragen-interleaved
--output-directory <...>
```

Sample Sheet

A sample sheet (`SampleSheet.csv`) records information about samples, the corresponding indexes, and other information that dictates the behavior of DRAGEN. The default location of the sample sheet is the input folder. To specify any CSV file in any location, use the command `--sample-sheet`. When a sample sheet does not exist in the default location and no sample sheet is specified in the command line, DRAGEN produces an error unless the `--no-sample-sheet true` option is specified (provided for legacy applications with no demultiplexing, adapter trimming, or other sample-sheet-specified settings supported).

In addition to the command line options that control the behavior of BCL conversion, use the [Settings] section in the sample sheet configuration file to specify how the samples are processed. The following are the sample sheet settings for BCL conversion.

Sample Sheet Versions

DRAGEN supports both sample sheets v1 and v2. The following table displays the different supported options for v1 and v2.

Sample Sheet v1	Sample Sheet v2
Supports both [Settings] and [settings]. Neither are required.	Supports only [BCLConvert_Settings]. Required.
Unrecognized settings trigger a warning.	Unrecognized settings produce an error and analysis aborts.

Settings Section

In addition to the command line options that control the behavior of BCL conversion, you can use the [Settings] section in the sample sheet configuration file to specify how the samples are processed. The following are the sample sheet settings for BCL conversion.

i | DRAGEN does not support the following sample sheet settings from bcl2fastq:

- ReverseComplement

Setting	Default	Value	Description
AdapterBehavior	trim	trim, mask	Whether the adapter should be trimmed or masked.
AdapterRead1	Not applicable	Read 1 adapter sequence containing A, C, G, or T.	The sequence to trim or mask from the end of Read 1. This option can only be specified if the first genomic read is included according to RunInfo.xml 1 or OverrideCycles.

Setting	Default	Value	Description
AdapterRead2	Not applicable	Read 2 adapter sequence containing A, C, G, or T.	The sequence to trim or mask from the end of Read 2. This option can only be specified if the second genomic read is included according to RunInfo.xml or OverrideCycles.
AdapterStringency	0.9	Float between 0.5 and 1.0	The stringency for matching the read to the adapter using the sliding window algorithm. This option can only be specified if AdapterRead 1 or AdapterRead 2 is specified.

Setting	Default	Value	Description
BarcodeMismatchesIndex1	1	0, 1, or 2	The number of allowed mismatches between the first Index Read and index sequence. Can only be specified when index1 is present and used for demultiplexing for all samples according to the index column and the OverrideCycles setting.

Setting	Default	Value	Description
BarcodeMismatchesIndex	1 2	0, 1, or 2	The number of allowed mismatches between the second Index Read and index sequence. Can only be specified when index2 is present and used for demultiplexing for all samples according to the index2 column and the OverrideCycles setting.

Setting	Default	Value	Description
MinimumTrimmedReadLength	The minimum of 35 and the shortest non-indexed read length.	0 to the shortest non-indexed read length	Reads trimmed below this point become masked at that point. Can only be specified when index2 is present and used for demultiplexing for all samples according to the index column and the OverrideCycles setting.
MinimumAdapterOverlap	1	1, 2, or 3	Do not trim detected adapter sequences shorter than this value.
MaskShortReads	The minimum of 22 and MinimumTrimmedReadLength.	0 to MinimumTrimmedReadLength	Reads trimmed below this point become masked out.

Setting	Default	Value	Description
OverrideCycles	None	Y: Specifies a sequencing read I: Specifies an indexing read U: Specifies a UMI length to be trimmed from read	String used to specify UMI cycles and mask out cycles of a read.
TrimUMI	true	true or false (1 or 0)	If set to false, UMI sequences are not trimmed from output FASTQ reads. The UMI is still placed in sequence header. Can only be enabled if a UMI is present for at least 1 sample according to the OverrideCycles setting.

Setting	Default	Value	Description
CreateFastqForIndexReads	0	true or false (1 or 0)	If set to true, output FASTQ files for index reads as well as genomic reads. Can only be enabled when an index is present and used for demultiplexing according to the index/index2 columns and the OverrideCycles setting.
NoLaneSplitting	false	true or false	If set to true, output all lanes of a flow cell to the same FASTQ files consecutively.

Setting	Default	Value	Description
FastqCompressionFormat	gzip	gzip, dragen, dragen-interleaved	Define the compression format: If the value is gzip, output FASTQ.GZ. If the value is dragen, output FASTQ.ORA not interleaving paired reads. If the value is dragen- interleaved, output FASTQ.ORA interleaving paired reads in a single FASTQ.ORA file for a higher compression rate. Specify the directory of the DRAGEN ORA reference files using the --ora- reference command. This can also be controlled using the command line options.

Setting	Default	Value	Description
FindAdaptersWithIndels	false	true or false	Use single-indel-detection adapter trimming (for matching default bcl2fastq2 behavior)

Setting	Default	Value	Description
IndependentIndexCollisionCheck	empty	Integer between 1 and the number of lanes that exist according to the RunInfo.xml.	Semi-colon-separated list of lanes which will use stricter validation. When enabled for any given lane, a barcode collision among samples in the corresponding lane(s) will be identified if at least one index (index or index2) have a collision. When disabled (default), a barcode collision among samples in the corresponding lane(s) will be identified if both indexes (index and index2) have a collision.

Override Cycles

The OverrideCycles mask elements are semicolon separated. For example:

```
OverrideCycles,U7N1Y143;I8;I8;U7N1Y143
```

DRAGEN supports flexible UMI processing during BCL conversion to support more third-party assays, including UMI sequences in index reads and multiple UMI regions per read. UMI sequences are trimmed from FASTQ read sequences and placed in the sequence identifier for each read, as normal.

The following are examples of OverrideCycles settings using 2x151 reads:

Setting	Description
OverrideCycles,U7N1Y143;I8;I8;U7N1Y143	UMI is composed of the first 7 bp of each genomic read, linked by 1 bp of ignored sequence, and is the format for Illumina nonrandom UMIs, used in the following products: <ul style="list-style-type: none"> • TruSight Oncology 170 RUO • TruSight Oncology 500 RUO • IDT for Illumina - UMI Index Anchors
OverrideCycles,Y151;I8;U10;Y151	Index Read 2 is a 10 bp UMI, and is the format for Agilent XT HS.
OverrideCycles,Y151;I8U9;I8;Y151	Index Read 1 contains both an index and a 9 bp UMI, and is the format for IDT Dual Index Adapters with UMIs.
OverrideCycles,U3N2Y146;I8;I8;U3N2Y146	UMI is composed of the first 3 bp of each genomic read, linked by 2 bp of ignored sequence, and is the format for UMIs in SureSelect XT HS 2 and IDT xGen Duplex Seq Adapter.
OverrideCycles,Y151;I8;I8;U10N12Y127	UMI is at the beginning of Read 2, attached with a linker sequence of length 12.

No lane splitting

When using `--no-lane-splitting true` or the corresponding sample sheet setting `NoLaneSplitting=true`, DRAGEN FASTQ file name convention and FASTQ contents match `bcl2fastq2` for the same feature.

DRAGEN only supports this mode when the `Lane` column is specified in the sample sheet to make sure that all samples are present in all lanes in the same order listed. This order is expected for flow cells with no fluidic boundaries between lanes.

IndependentIndexCollisionCheck

When this mode is enabled for a lane, each index (i7 & i5) must resolve to a single barcode within mismatch tolerances, because ambiguities can be resolved by the other index by default.

Combinatorial (exact) matches are still allowed. Refer to [Demultiplexing on page 639](#) for more information.

Data Section

The data section is required. Headers for the data section should be [Data] or [data] for sample sheet v1 and [BCLConvert_Data] for sample sheet v2. DRAGEN uses columns in the Data section to sort samples and index adapters.

Column	Description
Lane	When specified, DRAGEN generates FASTQ files only for the samples with the specified lane number. Only one valid integer is allowed, as defined by the <code>RunInfo.xml</code> .
Sample_ID	The sample ID.
index	The Index1 (i7) index adapter sequence. The length of string must match the number of first index cycles in <code>RunInfo.xml</code> or number specified in <code>OverrideCycles</code> . Reverse-complement of listed sequence is used if <code>RunInfo</code> has an <code>IsReverseComplement</code> tag with value <code>Y</code> .
index2	The Index2 (i5) Index adapter sequence. The length of string must match number of second index cycles in <code>RunInfo.xml</code> or number specified in <code>OverrideCycles</code> . Reverse-complement of listed sequence is used if <code>RunInfo</code> has an <code>IsReverseComplement</code> tag with value <code>Y</code> .
Sample_Project	Can only contain alphanumeric characters, dashes, and underscores. Duplicate data strings with different cases (eg, <code>sampleProject</code> and <code>SampleProject</code>) are not allowed. If these data strings are used, analysis fails. This column is not used unless you are using the command line option <code>--bcl-sampleproject-subdirectories</code> . See Command Line Options on page 615 for more information on command line options.
Sample_Name	If present, and both <code>--sample-name-column-enabled true</code> and <code>--bcl-sampleproject-subdirectories true</code> command lines are used, then output FASTQ files to subdirectories based upon <code>Sample_Project</code> and <code>Sample_ID</code> , and name FASTQ files by <code>Sample_Name</code>

Per Sample Settings

DRAGEN/bcl-convert 4.1 and later supports the following settings as columns in the `BCLConvert_Data` section, allowing them to be specified differently for each sample:

- `OverrideCycles`
- `BarcodeMismatchesIndex1`
- `BarcodeMismatchesIndex2`
- `AdapterRead1`
- `AdapterRead2`
- `AdapterBehavior`
- `AdapterStringency`

The per-sample settings can be specified by omitting the setting from the `BCLConvert_Settings` section, and instead adding a column to the `BCLConvert_Data` section with the setting name. Settings that do not apply to a sample, e.g `index2` if `i5` is masked out for that sample, must be blank or `na` in the entry for that sample.

This feature is only supported on version two `v2` sample sheets, and setting cannot be specified both globally and per-sample. Specifying `OverrideCycles` differently per-sample allows mixing of different pools into the same lane, but must follow barcode mismatch constraints for all cycles that are used for demultiplexing by any sample in that lane. DRAGEN software will detect all conflicts between samples at the beginning of the conversion run, even between different pools.

Different strategies, such as UMI indexes and dual-index inputs, can be combined if `IndependentIndexCollisionCheck` is not enabled.

The following is an example sample sheet using per-sample-settings:

```
[Header] FileFormatVersion,2
[BCLConvert_Settings] AdapterRead1,AGATCGGAAGAGCACACGTCTGAACCTCCAGTCA
AdapterRead2,AGATCGGAAGAGCGTCGTAGGGAAAGAGTGT
[BCLConvert_Data] Sample_ID,index,index2,OverrideCycles
21599,ATAGAGGC,TATAGCCT,Y151;I8;I8;Y151 21600,na,ATAGAGGC,Y151;U8;I8;Y151
21601,GGCTCTG,CCTATCC,Y151;I7N1;I7N1;Y151
21602,ATTACTCG,GGCTCTGA,Y151;I8;I8;U10Y141
```

Sample Sheet Obsolete Settings

DRAGEN does not support the following settings, and new formats must replace their corresponding old formats, when applicable. Manual changes to the sample sheet can be made to the `[Settings]` section, but the `[Data]` section must remain unchanged. If any of the obsolete settings are used in the command line or the sample sheet, DRAGEN aborts and returns an error. Also note that some obsolete settings that were previously specified on the command line are now correctly specified in the sample

sheet.

Table 10 Adapter Behavior and Specifications

Behavior	Obsolete Settings	New Settings
Designate the adapter sequence for Read 1 and Read 2 and specify the behavior as trim.	(sample sheet) Adapter, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA OR TrimAdapter, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA	(sample sheet) AdapterRead1, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA AND AdapterRead2, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA
Designate the same adapter sequence for Read 1 and Read 2 and specify the behavior as mask.	(sample sheet) MaskAdapter, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA	(sample sheet) AdapterRead1, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA AND AdapterRead2, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA AND AdapterBehavior, mask

Behavior	Obsolete Settings	New Settings
Designate the adapter sequence for Read 1 and Read 2 and specify the behavior as mask.	(sample sheet) MaskAdapter, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA OR MaskAdapterRead2, AGATCGGAAGAGCGTCGTAGGGAA AGAGTGT	(sample sheet) AdapterRead1, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA AND AdapterRead2, AGATCGGAAGAGCGTCGTAGGGAA AGAGTGT AND AdapterBehavior, mask
Designate the adapter sequence for Read 1 and Read 2 and specify the behavior as trim. Also specify 0.5 as the adapter stringency.	(sample sheet) Adapter, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA OR TrimAdapter, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA (command line) --adapter-stringency 0.5	(sample sheet) AdapterRead1, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA AND AdapterRead2, AGATCGGAAGAGCACACGTCTGAACTC CAGTCA(sample sheet) AND AdapterStringency, 0.5

Table 11 Read Trimming

Behavior	Obsolete Settings	New Settings
Trim the first 7 bases and last 6 bases of Read 1 for a 151 x 8 x 8 x 151 run.	(sample sheet) Read1StartFromCycle, 8 Read1EndWithCycle, 145	(sample sheet) N7Y137N6;I8;I8;Y151

Table 12 UMI Specification

	Obsolete Settings	New Settings
Designate the first 8 cycles of Read 1 and Read 2 as UMIs and trim the trailing base for a 151 x 8 x 8 x 151 run.	(sample sheet) Read1UMIStartFromCycle, 1 Read1UMLength, 8 Read1StartFromCycle, 10 Read2UMIStartFromCycle, 1 Read2UMLength, 8 Read2StartFromCycle, 10	(sample sheet) U8N1Y142;I8;I8;U8N1Y142

Table 13 Barcode Mismatches

Behavior	Obsolete Command Line Settings	New Sample Sheet Settings
Allow 1 mismatch in the i7 index sequence and 1 mismatch i5 index sequence.	--barcode-mismatches 1 OR --barcode-mismatches 1,1	BarcodeMismatchesIndex1, 1 AND BarcodeMismatchesIndex2, 1
Allow 2 mismatches in the i7 index sequence and 2 mismatches in the i5 index sequence.	--barcode-mismatches 2 OR --barcode-mismatches 2,2	BarcodeMismatchesIndex1, 2 AND BarcodeMismatchesIndex2, 2

Table 14 Masking of Trimmed Reads

Behavior	Obsolete Command Line Settings	New Sample Sheet Settings
Make sure that all trimmed reads are at least 10 base pairs long after adapter trimming by appending Ns to any read shorter than 10 base pairs.	--minimum-trimmed-read-length 10	MinimumTrimmedReadLength, 10
Make sure that all trimmed reads below 5 base pairs long are masked with Ns.	--mask-short-adapter-reads, 5	MaskShortReads, 5

Run Instructions

The following information has some additional instructions for running BCL Convert and DRAGEN for BCL conversion.

nohup

It is recommended to use `nohup`, or another protection, when executing BCL conversion via the command line in order to prevent a disconnection or terminal closure from terminating the process. This is performed by beginning the command line with `nohup` before the executable you wish to run.

Ulimit Settings

BCL Convert requires high ulimit settings for both the number of open files allowed and maximum user processes. If a run fails due to maximum user processes being set too low, an error message stating "resource temporarily unavailable" occurs. By default, BCL Convert attempts to set the ulimit soft limit for the number of open files `ulimit -n` to 65535 and the maximum user processes to 32768. If the values exceed the hard limits of the system, the soft limit is set to the hard limit. If more than 10,000 samples are provided, then `ulimit -n` is set to 720000.

Missing File Handling

If `--strict-mode` is set to `false`, BCL Convert executes certain behaviors when it finds missing or corrupt files, rather than abort operation. The following are the possible behaviors according to file type and status.

File type	Status	Behavior
*.bcl	Missing or corrupt	All base calls of the cycle in the corresponding lane and tile are replaced with N and a quality score of #.
*.bcl	Missing or corrupt	All base calls of the cycle in the corresponding lane and surface are replaced with N and a quality score of #.
*.locs	Missing or corrupt	Produce FASTQ files with artificial position data for all reads in the corresponding lane and tiles.
*.filter	Missing or corrupt	No FASTQ entries produced for any reads in the corresponding lane and tiles.
*.bci lane	Missing or corrupt	No FASTQ entries produced for any reads in the corresponding lane and tiles.

Analysis Methods

DRAGEN Bio-It Platform performs the following analysis:

Demultiplexing

DRAGEN produces one FASTQ file for each sample for each lane and read. Demultiplexing behaviors are as follows.

- When a sample sheet contains multiplexed samples, DRAGEN performs the following:
 - Places the reads without a matching index adapter sequence in `Undetermined_S0.fastq`.
 - Places the reads with valid index adapter sequences in the sample FASTQ file.
- When a sample sheet contains one unindexed sample, all reads are placed in the sample FASTQ files (one each for Read 1 and Read 2).
- All reads that do not demultiplex to the samples defined in the Data section of the sample sheet are placed in `Undetermined_S0.fastq` per lane.
- When the Lane column in the Data section is not used, all lanes are converted. Otherwise, only populated lanes are converted.

Reverse Complement

BCL Convert demultiplexes indices according to the orientation in which the sequencer evaluated the index read(s). The `IsReverseComplement` flag in the object of the `RunInfo.xml` specifies whether each index read is sequenced in the forward or reverse orientation. The flag may not be present depending on the sequencing instrument. If the `IsReverseComplement` option is present, BCL Convert will interpret the index sequences as specified.

- If the index read was sequenced in the forward orientation, `IsReverseComplement` flag will be specified as `N`.
- If the index read was sequenced in the reverse orientation, `IsReverseComplement` flag will be specified as `Y`, and the following steps will be taken by BCL Convert:
 - The software will reverse the sequence and generate the complement base pair as the index read, where $A=T$, $C=G$, and $N=N$.
 - The `OverrideCycles` value specified will be reversed for the corresponding index read before the reverse complement is taken.
 - If a UMI is specified in the corresponding index read, the `r` character will be added at the beginning of the UMI sequence written in the Read Name of the FASTQ file.
 - A log message will be displayed indicating that the reverse complement was used for the corresponding index read.

Combined vs Independent Index Validation

DRAGEN/bcl-convert version 3.10 introduced a stricter barcode validation system that required each index in a dual-index setup to independently resolve against other samples at the index's mismatch tolerance, rather than allowing the combination of indexes to resolve in the case of a conflict in i7 or i5

individually. It was incompatible with previous versions of DRAGEN/bcl-convert and with bcl2fastq2, but is a more strict validation that may be better suited to the accuracy requirements of unique-dual applications.

DRAGEN/bcl-convert 4.1.0 introduced a specified per lane option to enable a more relaxed validation compatible with bcl2fastq2 and earlier versions of DRAGEN/bcl-convert. The `CombinedIndexCollisionCheck` setting enabled relaxed validation on a per-lane basis.

In DRAGEN/bcl-convert 4.1.6 relaxed validation was the default behavior and a setting to enable stricter validation was added. The `IndependentIndexCollisionCheck` setting enables strict validation on the given semi-colon-separated list of lanes.

The following example sets lanes 1, 3, & 4 to strict validation mode:

```
[BCLConvert_Settings] CombinedIndexCollisionCheck,1;3;4
```

 DRAGEN/bcl-convert 4.1.6+ does not support the `CombinedIndexCollisionCheck` setting and will produce an error if used.

UMI Trimming

DRAGEN is capable of trimming unique molecular identifier (UMI) sequences from the genomic or index sequences. The cycles of the sequencing read that correspond to the UMI are specified in the `OverrideCycles` parameter in the Settings section of the sample sheet. See the [Settings Section on page 622](#) to set the `OverrideCycles` parameter.

The following are details of the behavior of reads specified as UMIs:

- UMIs are trimmed from the sequence by default. Use the `TrimUMI` setting in the sample sheet to include UMIs.
- UMI sequence can be specified in the index and genomic reads. More than one UMI sequence can be specified per read.
- The specified UMI cycles are applied to all clusters. There is no mechanism to apply UMI based on lane or sample.
- UMI sequences can only be specified at the beginning and end of sequencing and index reads. UMIs cannot be located in the middle of a read.

Adapter Trimming and Masking

DRAGEN can mask or trim adapter sequences from read data so that those adapter sequences are not passed to any downstream analysis steps.

Additional details of the adapter handling capabilities are as follows.

- DRAGEN masks the identified adapter sequence with N so that the overall read length is constant across all clusters in the read.

- DRAGEN trims the identified adapter sequence from the read. The length for each cluster varies due to trimming.
- DRAGEN assumes that input adapter sequences can only contain A, C, G, or T.

Output Files

FASTQ Files

As converted versions of BCL files, FASTQ files are the primary output of BCL Convert. Like BCL files, FASTQ files contain base calls with associated Q-scores. Unlike BCL files, which contain per-cycle data, FASTQ files contain the per-read data that most analysis applications require.

The software generates one FASTQ file for every sample, read, and lane. For example, for each sample in a paired-end run, the software generates two FASTQ files: one for Read 1 and one for Read 2. In addition to these sample FASTQ files, the software generates two FASTQ files per lane containing all unknown samples. FASTQ files for Index Read 1 and Index Read 2 are not generated because the sequence is included in the header of each FASTQ entry.

- If `Sample_Name` and `Sample_Project` are both present, and both `--sample-name-column-enabled true` and `--bcl-sampleproject-subdirectories true` command lines are used, then the output FASTQ files to subdirectories based on `Sample_Project` and `Sample_ID`, and name fastq files by `Sample_Name`. The same project directory contains the files for multiple samples.
- If the `Sample_ID` and `Sample_Name` columns are specified but do not match, the FASTQ files reside in a subdirectory where files use the `Sample_Name` value.
- Reads with unidentified index adapters are recorded in one file named `Undetermined_S0_`. If a sample sheet includes multiple samples without specified index adapters, the software displays a missing barcode error and ends the analysis.

i | The software allows one unindexed sample since identification is not necessary to sequence one sample. Sequencing multiple samples requires multiplexing so the samples can be identified for analysis.

File Names

The file name format is constructed from fields specified in the sample sheet, using the format: <Sample_ID>_S#_L00#_R#_001.fastq.gz.

Example: <Sample_ID>_S1_L001_R1_001.fastq.gz

- <Sample_ID>: The ID of the sample provided in the sample sheet.
- S1: The number of the sample based on the order that samples are listed in the sample sheet, starting with 1. In the example, S1 indicates that the sample is the first sample listed for the run.

i | Reads that cannot be assigned to any sample are written to a FASTQ file as sample number 0 and excluded from downstream analysis.

- L001: The lane number of the flow cell, starting with lane 1, to the number of lanes supported.
- R1: The read. R1 indicates Read 1. R2 would indicate Read 2 of a paired-end run.
- 001: The last portion of the file name is always 001.

File Format

FASTQ files are text-based files that contain base calls with corresponding Q-scores for each read.

Each file has one 4-line entry:

- A sequence identifier with information about the run and cluster, formatted as:

```
@Instrument:RunID:FlowCellID:Lane:Tile:X:Y:UMI Read:Filter:0:IndexSequence or  
SampleNumber
```

i | If a UMI is specified in an index read when `isReverseComplement` exists in the `RunInfo.xml`, the `r` character will be added at the beginning of the UMI sequence written in the Read Name of the FASTQ file.

- The sequence (base calls A, G, C, T, and N, for unknown bases).
- A plus sign (+) that functions as a separator.
- The Q-score using ASCII 33 encoding. See [Quality Output File on page 644](#) for more information.

Sequence Identifier Fields

Field	Description
@	Each sequence identifier line starts with @.
instrument	The instrument ID.
run ID	The run number on the system.
flow cell ID	The flow cell ID.
lane	The flow cell lane number.
tile	The flow cell tile number.
x_pos	The X coordinate of the cluster.
y_pos	The Y coordinate of the cluster.
UMI	Optional. The UMI sequence (A, G, C, T, and N). When the sample sheet specifies UMIs, a plus sign separates the Read 1 and Read 2 sequences.

Field	Description
read	1 - Read 1, which is the first read of a paired-end run or the only read of a single-read run. 2 - Read 2, which is the second read of a paired-end run.
is filtered	N - No failed reads are included.
control number	0 - Control bits are not turned on.
index sequence or sample number	The Index Read sequence (A, G, C, T, and N). If the sample sheet indicates indexing, the index adapter sequence is appended to the end of the read identifier. If indexing is not indicated (one sample per lane), the sample number is appended to the read identifier.

A complete FASTQ file entry resembles the following example:

Log Files

BCL conversion outputs log files to the Logs/ output subfolder. These include three separate files, Info.log, Warnings.log, and Errors.log, for three increasing levels of severity. All output to these files is also written to the terminal console: Info is written to standard-out, while Errors and Warnings are written to standard-error.

In addition, the file "FastqComplete.txt" is created in the Logs/ subfolder when conversion is complete. This can be used to trigger subsequent action if desired.

BCL Metrics

DRAGENAnalysis app produces the following metrics in CSV format to the Reports / output subfolder. In addition, the sample sheet and RunInfo.xml file used during conversion is copied into the Reports / output subfolder.

Demultiplex Output File

The following metrics are included in the Demultiplex_Stats.csv output file.

Column	Description
Index	The contents of <code>index</code> in sample sheet for this sample. For dual-index, the value concatenated with <code>index2</code> .

Column	Description
# Reads	The total number of pass-filter reads mapping to this sample for the lane.
# Perfect Index Reads	The number of mapped reads with barcodes that match the indexes provided in the sample sheet.
# One Mismatch Index Reads	The number of mapped reads with barcodes matched with one base mismatched.
# Two Mismatch Index Reads	The number of mapped reads with barcodes matched with exactly two bases mismatched.
% Reads	The percentage of pass-filter reads mapping to this sample for the lane.
% Perfect Index Reads	The percentage of mapped reads with barcodes that match the indexes provided in the sample sheet exactly.
% One Index Reads	The percentage of mapped reads with barcodes matched with exactly one base mismatched.
% Two Index Reads	The percentage of mapped reads with barcodes matched with exactly two bases mismatched.

Quality Output File

The following metrics are included in the `Quality_Metrics.csv` output file.

Column	Description
Lane	The lane number that this metric line refers to.
Sample_ID	The contents of Sample_ID in the sample sheet for this sample.
index	The contents of Index in sample sheet for this sample.
index2	The contents of Index 2 in the sample sheet for this sample.
ReadNumber	The read number this metric line refers to.
Yield	The total number of bases mapping to the sample in this read.
YieldQ30	The total number of bases with quality score ≥ 30 mapping to the sample in this read.
QualityScoreSum	The sum of quality scores of bases mapping to the sample in this read.
Mean Quality Score (PF)	The mean quality score of bases mapping to the sample in this read.
% Q30	The percentage of bases with quality score ≥ 30 mapping to the sample in this read.

Adapter Output File

The following information is included in the `Adapter_Metrics.csv` output file.

Column	Description
Lane	The lane number this metric line refers to.
Sample_ID	The contents of Sample_ID in the sample sheet for this sample
index	The contents of Index 1 (i7) in sample sheet for this sample.
index2	The contents of Index 2 (i5) in the sample sheet for this sample.
ReadNumber	The read number this metric line refers to.
AdapterBases	The total number of bases trimmed as adapter from the read in the sample.
SampleBases	The total number of bases not trimmed from the read in the sample.
% Adapter Bases	The percentage of bases trimmed as adapter from the read in the sample.

Index Hopping Output File

For unique dual index inputs, the `Index_Hopping_Counts.csv` file provides the number of reads mapping to every combination of provided index and index2 values, including via mismatch tolerance. The metrics provide visibility into any index-hopping behavior that have occurred. The samples with both index and index2 values present in the sample sheet are present in the index hopping file. The following information is included in the `Index_Hopping_Counts.csv` output file.

Column	Description
Lane	The lane for each metric.
SampleID	If the index combination corresponds to a sample, the contents of Sample_ID in the sample sheet for this sample.
index	The contents of index in sample sheet for the sample.
index2	The contents of index 2 in sample sheet for the sample.
# Reads	The total number of pass-filter reads mapping to the index and index2 combination.
% of Hopped Reads	The percentage of hopped pass-filter reads mapping to the index and index2 combination.
% of All Reads	The percentage of all pass-filter reads mapping to the index and index2 combination.

Top Unknown Barcodes Output File

The `Top_Unknown_Barcodes.csv` file lists the most commonly encountered barcode sequences in the flow cell input that are not listed in the sample sheet. The 1,000 most common unlisted sequences are listed, along with any other sequences with a frequency equivalent to the 1,000th most commonly encountered sequence. The following information is included in the `Top_Unknown_Barcodes.csv` output file.

Column	Description
Lane	The lane for each metric.
index	The first index value of this unlisted sequence.
index2	The second index value of this unlisted sequence.
# Reads	The total number of pass-filter reads mapping to the index and index2 combination.
% of Unknown Barcodes	The percentage of unknown pass-filter reads mapping to the index and index2 combination.
% of All Reads	The percentage of all pass-filter reads mapping to the index and index2 combination.

Per-cycle Adapter Metrics

The following information is included in the `Adapter_Cycle_Metrics.csv` output file.

Column	Description
Lane	The lane number this metric line refers to.
Sample_ID	The contents of Sample_ID in the sample sheet for this sample.
index	The contents of index in sample sheet for this sample.
index2	The contents of index2 in the sample sheet for this sample.
ReadNumber	The read number this metric line refers to.
Cycle	The cycle number this metric line refers to.
NumClustersWithAdapterAtCycle	The number of clusters where the adapter was detected to begin precisely at this cycle.
% At Cycle	The percentage of all clusters where the adapter was detected to begin precisely at this cycle.

Per-tile Metrics

The format of Demultiplex_Tile_Stats.csv and Quality_Tile_Metrics.csv matches that of Demultiplex_Stats.csv and Quality_Metrics.csv, save that an additional column is added:

Column	Description
Tile	The tile numeral value this metric line refers to.

These files provide per-tile data rather than aggregated across the lane and read.

Sample_Name and Sample_Project Columns

For the metrics files listed above (apart from Top_Unknown_Barcodes.csv), up to two additional columns may be added to each line if 'bcl-sampleproject-subdirectories' and/or 'sample-name-column-enabled' options are enabled:

Column	Description
Sample_Project	The Sample_Project value for the sample this metric line refers to.
Sample_Name	The Sample_Name value for the sample this metric line refers to.

These files provide per-tile data rather than aggregated across the lane and read.

FASTQ Output File

The fastq_list.csv output file is located in the output folder with the FASTQ files, and provides the associations between the sample indexes, lane, and the output FASTQ file names. The columns of each row are shown, along with example entries from a test run. For more information on running DRAGEN using fastq_list.csv, see [FASTQ CSV File Format](#).

The following columns are provided per unique sample_ID and lane combination:

Column	Description
RGID	Read Group
RGSM	Sample ID
RGLB	Library
Lane	Flow cell lane
Read1File	Full path to a valid FASTQ input file
Read2File	Full path to a valid FASTQ input file. Required for paired-end input. If not using paired-end input, leave empty,

The following is an example fastq_list.csv output file.

```

RGID,RGSM,RGLB,Lane,Read1File,Read2File
AACAAACCA.ACTGCATA.1,1,UnknownLibrary,1,/home/user/dragen_bcl_out/1_S1_L001_R1_
001.fastq.gz,/home/user/dragen_bcl_out/1_S1_L001_R2_001.fastq.gz
AATCCGTC.ACTGCATA.1,2,UnknownLibrary,1,/home/user/dragen_bcl_out/2_S2_L001_R1_
001.fastq.gz,/home/user/dragen_bcl_out/2_S2_L001_R2_001.fastq.gz
CGAAACTTA.GCGTAAGA.1,3,UnknownLibrary,1,/home/user/dragen_bcl_out/3_S3_L001_R1_
001.fastq.gz,/home/user/dragen_bcl_out/3_S3_L001_R2_001.fastq.gz
GATAGACA.GCGTAAGA.1,4,UnknownLibrary,1,/home/user/dragen_bcl_out/4_S4_L001_R1_
001.fastq.gz,/home/user/dragen_bcl_out/4_S4_L001_R2_001.fastq.gz

```

Legacy Stats Output Files

When the `output-legacy-stats` command line option is enabled, DRAGENBCL Convert produces the following metrics to the Reports/legacy output subfolder. The files are identical to the `bcl2fastq2.20` report files except for incidences where there is decreased accuracy, non-deterministic output, or incorrect output from `bcl2fastq2.20`.

ConversionStats File

The ConversionStats.xml file contains the lane number for each lane and the following information for each tile:

- Raw Cluster Count Read Number
- YieldQ30
- Yield
- QualityScore Sum

DemultiplexStats File

The DemultiplexingStats.xml contains the flow cell ID and project name. For each sample, index, and lane, the file lists the BarcodeCount, PerfectBarcodeCount, and OneMismatchBarcodeCount (if applicable).

Adapter Trimming File

The adapter trimming file is a text-based file that contains a statistics summary of adapter trimming for a FASTQ file. The file contains the fraction of reads with untrimmed bases for each sample, lane, and read number plus the following information:

- Lane
- Read
- Project

- Sample ID
- Sample Name
- Sample Number
- TrimmedBases
- PercentageOfBases(beingtrimmed)

FastqSummaryF1L# File

A FastqSummaryF1L#.txt file contains the number of raw and passed filter reads for each sample and tile in a lane. The number sign (#) indicates the lane number.

DemuxSummaryF1L# File

DemuxSummaryF1L#.txt files, where # indicates the lane number, are generated when the sample sheet contains at least one indexed sample. A file contains the percentage of each tile that each sample occupies. It also lists the 1000 most common unknown index adapter sequences and the total number of reads with each index adapter identified.

i | To improve processing speed, the total for each index adapter is based on an estimate from a sampling algorithm.

HTML Reports

HTML reports are generated from data in DemultiplexingStats.xml and ConversionStats.xml. The reports reside in Reports\html in the output directory or in the directory specified by the --reports-dir option.

The flow cell summary contains the following information:

- Clusters(Raw) Clusters(PF)* Yield (MBases)

For patterned flow cells, the number of raw clusters is equal to the number of wells on the flow cell.

The lane summary provides the following information for each project, sample, and index sequence specified in the sample sheet:

- Lane#
- Clusters(Raw)
- %oftheLane
- % Perfect Barcode
- % One Mismatch
- Clusters(Filtered)
- Yield

- % PF Clusters
- %Q30Bases
- Mean Quality Score
- The Top Unknown Barcodes table in the HTML report provides the count and sequence for the 10 most common unmapped index adapters in each lane.

Monitoring System Health

When you power up your DRAGEN system a daemon (*dragen_mond*) is started that monitors the card for hardware issues. This daemon is also started when your DRAGEN system is installed or updated. The main purpose of this daemon is to monitor DRAGEN Bio-IT Processor temperature and abort DRAGEN when the temperature exceeds a configured threshold.

To manually start, stop, or restart the monitor, run the following as root:

```
sudo service dragen_mond [stop|start|restart]
```

By default, the monitor polls for hardware issues once per minute and logs temperature once every hour.

The */etc/sysconfig/dragen_mond* file specifies the command line options used to start *dragen_mond* when the service command is run. Edit DRAGEN_MOND_OPTS in this file to change the default options. For example, the following changes the poll time to 30 seconds and the log time to once every 2 hours:

```
DRAGEN_MOND_OPTS="-d -p 30 -l 7200"
```

The *-d* option is required to run the monitor as a daemon.

The *dragen_mond* command line options are as follows:

Option	Description
<i>-m</i> -- <i>swmaxtemp</i>	Maximum software alarm temperature (Celsius). Default is 85. <i><n></i>
<i>-i</i> -- <i>swmintemp</i>	Minimum software alarm temperature (Celsius). Default is 75. <i><n></i>
<i>-H</i> -- <i>hwmaxtemp</i>	Maximum hardware alarm temperature (Celsius). Default is 100. <i><n></i>

Option	Description
<code>-p --polltime <n></code>	Time between polling chip status register (seconds). Default is 60.
<code>-l --logtime <n></code>	Log FPGA temp every n seconds. Default is 3600. Must be a multiple of polltime
<code>-d --daemon</code>	Detach and run as a daemon.
<code>-h --help</code>	Print help and exit.
<code>-V --version</code>	Print the version and exit.

To display the current temperature of the DRAGEN Bio-IT Processor, use the `dragen_info -t` command. This command does not execute if `dragen_mond` is not running.

```
% dragen_info -t
FPGA Temperature: 42C (Max Temp: 49C, Min Temp: 39C)
```

Logging

All hardware events are logged to `/var/log/messages` and `/var/log/dragen_mond.log`. The following shows an example in `/var/log/messages` of a temperature alarm:

```
Jul 16 12:02:34 komodo dragen_mond[26956]: WARNING: FPGA software over temperature alarm has been triggered -- temp threshold: 85 (Chip status: 0x80000001)
Jul 16 12:02:34 komodo dragen_mond[26956]: Current FPGA temp: 86, Max temp: 88, Min temp: 48
Jul 16 12:02:34 komodo dragen_mond[26956]: All dragen processes will be stopped until alarm clears
Jul 16 12:02:34 komodo dragen_mond[26956]: Terminating dragen in process 1510 with SIGUSR2 signal
```

By default, temperature is logged to `/var/log/dragen_mond.log` every hour:

```
Aug 01 09:16:50 Setting FPGA hardware max temperature threshold to 100
Aug 01 09:16:50 Setting FPGA software max temperature threshold to 85
Aug 01 09:16:50 Setting FPGA software min temperature threshold to 75
Aug 01 09:16:50 FPGA temperatures will be logged every 3600 seconds
Aug 01 09:16:50 Current FPGA temperature is 52 (Max temp = 52, Min temp = 52)
Aug 01 10:16:50 Current FPGA temperature is 53 (Max temp = 56, Min temp = 49)
Aug 01 11:16:50 Current FPGA temperature is 54 (Max temp = 56, Min temp = 49)
```

If DRAGEN is executing when a thermal alarm is detected, the following is displayed in the terminal window of the DRAGEN process:

```
*****
** Received external signal -- aborting dragen. **
** An issue has been detected with the dragen card. **
** Check /var/log/messages for details. **
```

**

**

** It may take up to a minute to complete shutdown. **

If you see this message, stop running the DRAGEN software. Do the following to alleviate the overheating condition on the card:

- Be sure that there is ample air flow over the card. Consider moving the card to a slot where there is more air flow, adding another fan or increasing the fan speed.
- Give the card more space in the box. If there are available PCIe slots, move the card so that it has empty slots on either side.

Contact Illumina Technical Support if you are having trouble resolving the thermal alarm on your system.

Hardware Alarms

The following table lists the hardware events logged by the monitor when an alarm is triggered:

ID	Description	Monitor Action
0	Software overheating	Terminate usage until DRAGEN Bio-IT Processor cools to software minimum temperature.
1	Hardware overheating	Fatal. Aborts dragen software; system reboot required
2	Board SPD overheating	Logged as nonfatal
3	SODIMM overheating	Logged as nonfatal
4	Power 0	Fatal. Aborts dragen software; system reboot required
5	Power 1	Fatal. Aborts dragen software; system reboot required
6	DRAGEN Bio-IT Processor power	Logged as nonfatal
7	Fan 0	Logged as nonfatal
8	Fan 1	Logged as nonfatal
9	SE5338	Fatal. Aborts dragen software; system reboot required
10–30	Undefined (Reserved)	Fatal. Aborts dragen software; system reboot required

Fatal alarms prevent the DRAGEN host software from running and require a system reboot. When a software overheating alarm is triggered, the monitor looks for and aborts any running DRAGEN processes. The monitor continues to abort any new DRAGEN processes until the temperature decreases to the minimum threshold and the hardware clears the chip status alarm. When the software overheating alarm clears, DRAGEN jobs can resume executing.

Contact Illumina Technical Support with details from the log files if any of these alarms are triggered on your system.

Illumina Annotation Engine

Illumina Annotation Engine (IAE), also known as Nirvana, provides clinical-grade annotation of genomic variants, such as SNVs, MNVs, insertions, deletions, indels, STRs, SV, and CNVs. Use a VCF as input. The output is a structured JSON representation of all annotations and sample information extracted from the VCF. IAE can handle multiple alternate alleles and multiple samples.

VCF files can be enabled by annotation on the DRAGEN command-line or by running the standalone tool.

i Before running IAE, the external data sources, gene models, and reference genome need to be downloaded from the annotation server.

By default, the IAE binaries are located in the `/opt/edico/share/nirvana` directory. This directory includes two files: the Downloader and Nirvana (Illumina Annotation Engine).

Limitations

The Illumina Annotation Engine and the Downloader are compatible with the following platforms:

- CentOS 7 and other modern Linux distributions using x64 processors.

Download Data Files

To store annotation data files, create a top-level directory. The created directory contains three subdirectories:

- Cache contains gene models.
- SupplementaryAnnotation contains external data sources like dbSNP and gnomAD.
- References contains the reference genome.

The following command line options are used:

Option	Value	Example	Description
--ga	GRCh37, GRCh38, or both	GRCh38	Genome assembly

Option	Value	Example	Description
--out	Output directory	~/Data	Top-level output directory

Download data files as follows.

i If the Illumina DRAGEN Compute Server does not have an internet connection, the Downloader executable can be copied to a non-Illumina DRAGEN Compute Server that is connected to the internet to download the annotation data. Once the download has completed, the annotation data can then be copied locally to the Illumina DRAGEN Compute Server for subsequent annotation.

1. To create a data directory, enter the following command.

This example creates the Data directory in your home directory.

```
mkdir ~/Data
```

2. Download the files for a genome assembly.

This example downloads the genome assembly GRCh38.

```
/opt/edico/share/nirvana/Downloader --ga GRCh38 --out ~/Data
```

You can use the same command to resynchronize the data sources with the Illumina Annotation Engine servers, including the following actions:

- Remove obsolete files, such as old versions of data sources, from the output directory.
- Download newer files.

The following is the created output:

```
-----
Downloader (c) 2020 Illumina, Inc.
Stromberg, Roy, Lajugie, Jiang, Li, and Kang 3.9.1-0-gc823805
-----
- downloading manifest... 37 files.
- downloading file metadata:
- finished (00:00:00.8).
- downloading files (22.123 GB):
- downloading 1000_Genomes_Project_Phase_3_v3_plus_refMinor.rma.idx (GRCh38)
- downloading MITOMAP_20200224.nsa.idx (GRCh38)
- downloading ClinVar_20200302.nsa.idx (GRCh38)
- downloading REVEL_20160603.nsa.idx (GRCh38)
- downloading phyloP_hg38.npd.idx (GRCh38)
- downloading ClinGen_Dosage_Sensitivity_Map_20200131.nsi (GRCh38)
- downloading MITOMAP_SV_20200224.nsi (GRCh38)
- downloading dbSNP_151_globalMinor.nsa.idx (GRCh38)
- downloading ClinGen_Dosage_Sensitivity_Map_20190507.nga (GRCh38)
```

```

- downloading PrimateAI_0.2.nsa.idx (GRCh38)
- downloading ClinGen_disease_validity_curations_20191202.nga (GRCh38)
- downloading 1000_Genomes_Project_Phase_3_v3_plus.nsa.idx (GRCh38)
- downloading SpliceAi_1.3.nsa.idx (GRCh38)
- downloading dbSNP_153.nsa.idx (GRCh38)
- downloading TOPMed_freeze_5.nsa.idx (GRCh38)
- downloading MITOMAP_20200224.nsa (GRCh38)
- downloading gnomAD_2.1.nsa.idx (GRCh38)
- downloading ClinGen_20160414.nsi (GRCh38)
- downloading gnomAD_gene_scores_2.1.nga (GRCh38)
- downloading 1000_Genomes_Project_(SV)_Phase_3_v5a.nsi (GRCh38)
- downloading Multiz100Way_20171006.pcs (GRCh38)
- downloading 1000_Genomes_Project_Phase_3_v3_plus_refMinor.rma (GRCh38)
- downloading ClinVar_20200302.nsa (GRCh38)
- downloading OMIM_20200409.nga (GRCh38)
- downloading Both.transcripts.ndb (GRCh38)
- downloading REVEL_20160603.nsa (GRCh38)
- downloading PrimateAI_0.2.nsa (GRCh38)
- downloading dbSNP_151_globalMinor.nsa (GRCh38)
- downloading Both.sift.ndb (GRCh38)
- downloading Both.polyphen.ndb (GRCh38)
- downloading Homo_sapiens.GRCh38.Nirvana.dat
- downloading 1000_Genomes_Project_Phase_3_v3_plus.nsa (GRCh38)
- downloading phyloP_hg38.npd (GRCh38)
- downloading SpliceAi_1.3.nsa (GRCh38)
- downloading TOPMed_freeze_5.nsa (GRCh38)
- downloading dbSNP_153.nsa (GRCh38)
- downloading gnomAD_2.1.nsa (GRCh38)
- finished (00:04:10.1).

```

Description Status

```

1000_Genomes_Project_(SV)_Phase_3_v5a.nsi (GRCh38) OK
1000_Genomes_Project_Phase_3_v3_plus.nsa (GRCh38) OK
1000_Genomes_Project_Phase_3_v3_plus.nsa.idx (GRCh38) OK
1000_Genomes_Project_Phase_3_v3_plus_refMinor.rma (GRCh38) OK
1000_Genomes_Project_Phase_3_v3_plus_refMinor.rma.idx (... OK
Both.polyphen.ndb (GRCh38) OK
Both.sift.ndb (GRCh38) OK
Both.transcripts.ndb (GRCh38) OK

```

```
ClinGen_20160414.nsi (GRCh38) OK
ClinGen_Dosage_Sensitivity_Map_20190507.nga (GRCh38) OK
ClinGen_Dosage_Sensitivity_Map_20200131.nsi (GRCh38) OK
ClinGen_disease_validity_curations_20191202.nga (GRCh38) OK
ClinVar_20200302.nsa (GRCh38) OK
ClinVar_20200302.nsa.idx (GRCh38) OK
Homo_sapiens.GRCh38.Nirvana.dat OK
MITOMAP_20200224.nsa (GRCh38) OK
MITOMAP_20200224.nsa.idx (GRCh38) OK
MITOMAP_SV_20200224.nsi (GRCh38) OK
Multiz100Way_20171006.pcs (GRCh38) OK
OMIM_20200409.nga (GRCh38) OK
PrimateAI_0.2.nsa (GRCh38) OK
PrimateAI_0.2.nsa.idx (GRCh38) OK
REVEL_20160603.nsa (GRCh38) OK
REVEL_20160603.nsa.idx (GRCh38) OK
SpliceAi_1.3.nsa (GRCh38) OK
SpliceAi_1.3.nsa.idx (GRCh38) OK
TOPMed_freeze_5.nsa (GRCh38) OK
TOPMed_freeze_5.nsa.idx (GRCh38) OK
dbSNP_151_globalMinor.nsa (GRCh38) OK
dbSNP_151_globalMinor.nsa.idx (GRCh38) OK
dbSNP_153.nsa (GRCh38) OK
dbSNP_153.nsa.idx (GRCh38) OK
gnomAD_2.1.nsa (GRCh38) OK
gnomAD_2.1.nsa.idx (GRCh38) OK
gnomAD_gene_scores_2.1.nga (GRCh38) OK
phyloP_hg38.npd (GRCh38) OK
phyloP_hg38.npd.idx (GRCh38) OK
-----
Peak memory usage: 52.3 MB
Time: 00:04:12.2
```

Annotate Files (DRAGEN Command-Line)

Add the following command-line arguments to automatically annotate output VCF files:

Argument	Example	Description
--enable-variant-annotation	true	enables annotation if the pipeline supports it
--variant-annotation-data	/path/to/your/NirvanaData	the location where you downloaded the Nirvana annotation files
--variant-annotation-assembly	GRCh38	the genome assembly - either GRCh37 or GRCh38. hg19 is handled properly by using GRCh37

All the command-line arguments shown together:

```
--enable-variant-annotation=true --variant-annotation-data=/path/to/your/NirvanaData --variant-annotation-assembly=GRCh38
```

Annotate Files (Standalone Nirvana Tool)

1. If you have not generated a VCF file, download a VCF file using the following command

```
curl -O
https://raw.githubusercontent.com/HelixGrind/DotNetMisc/master/TestFiles/HiSeq.10000.vcf.gz
```

IAE supports uncompressed VCF files and bgzip compressed VCF files. VCF files that have been compressed by standard gzip are not supported.

2. To annotate the file, enter the following command:

```
/opt/edico/share/nirvana/Nirvana -c ~/Data/Cache/GRCh38/Both \ -r
~/Data/References/Homo_sapiens.GRCh38.Nirvana.dat \ --sd
~/Data/SupplementaryAnnotation/GRCh38 -i HiSeq.10000.vcf.gz -o HiSeq.10000
```

The following are the available command line options:

Option	Value	Example	Description
-c	Directory	~/Data/Cache/GRCh38/Both	Cache directory
-r	Directory	~/Data/References/Homo_sapiens.GRCh38.Nirvana.dat	Reference directory
--sd	Directory	~/Data/SupplementaryAnnotation/GRCh38	Supplementary annotation directory

Option	Value	Example	Description
-i	path	HiSeq.10000.vcf.gz	Input VCF path
-o	prefix	HiSeq.10000	Output path prefix

Using the example above, IAE generates the following output called HiSeq.10000.json.gz.

```
-----
Nirvana (c) 2020 Illumina, Inc.
Stromberg, Roy, Lajugie, Jiang, Li, and Kang 3.9.1-0-gc823805
-----
Initialization Time Positions/s
-----
Cache 00:00:01.9
SA Position Scan 00:00:00.4 23,867
Reference Preload Annotation Variants/s
-----
chr1 00:00:00.4 00:00:03.7 2,651
Summary Time Percent
-----
Initialization 00:00:02.3 25.7 %
Preload 00:00:00.4 5.4 %
Annotation 00:00:03.7 41.5 %
Peak memory usage: 1.284 GB
Time: 00:00:08.0
```

JSON Output file

IAE produces an output file in JSON format that includes the following three sections:

Section	Content
Header	Configuration, data source versions, and sample names.
Positions	Variant level annotation.
Genes	Gene level annotation.

The following are outputs using the example commands specified in the sections above. Each of the following sections contains only a subset of information. The output file contains more information.

Header

The following is an example of the Header section.

```
"header": {  
  "annotator": "Nirvana 3.9.0",  
  "creationTime": "2020-06-03 08:05:06",  
  "genomeAssembly": "GRCh38",  
  "schemaVersion": 6,  
  "dataVersion": "91.26.57",  
  "dataSources": [  
    {  
      "name": "VEP",  
      "version": "91",  
      "description": "BothRefSeqAndEnsembl",  
      "releaseDate": "2018-03-05"  
    },  
    {  
      "name": "ClinVar",  
      "version": "20200302",  
      "description": "A freely accessible, public archive of reports of the  
      relationships among human variations and phenotypes, with supporting evidence",  
      "releaseDate": "2020-03-02"  
    },  
    {  
      "name": "dbSNP",  
      "version": "153",  
      "description": "Identifiers for observed variants",  
      "releaseDate": "2019-07-22"  
    },  
    {  
      "name": "gnomAD",  
      "version": "2.1",  
      "description": "gnomAD allele frequency data remapped to GRCh38 with CrossMap  
      by Ensembl",  
      "releaseDate": "2019-03-25"  
    },  
    {  
      "name": "PrimateAI",  
      "version": "0.2",  
      "description": "PrimateAI percentile scores.",  
      "releaseDate": "2018-11-07"  
    },
```

```
{
  "name": "OMIM",
  "version": "20200409",
  "description": "An Online Catalog of Human Genes and Genetic Disorders",
  "releaseDate": "2020-04-09"
}
],
"samples": [
  "NA12878"
]
},
```

Positions

Each position represents one row of the VCF file. Each position contains a samples section and a variants section. Reference and alternate alleles shown here matches the VCF content exactly.

The samples section lists the sample-specific information, such as genotype, in the same order as they appear in the VCF and in the JSON header above.

The variants section provides annotations for each alternate allele that appear in the VCF row. This includes allele-specific annotation from external data sources as well as transcript-level annotation. Reference and alternate alleles shown here are shown in their most shortened representation. For example, padding bases have been removed and the variant are left-aligned.

```
"positions": [
{
  "chromosome": "chr1",
  "position": 1043248,
  "refAllele": "C",
  "altAlleles": [
    "T"
  ],
  "quality": 441.42,
  "filters": [
    "PASS"
  ],
  "strandBias": -425.94,
  "cytogeneticBand": "1p36.33",
  "samples": [
    {
```

```
"genotype": "0/1",
"variantFrequencies": [
0.537
],
"totalDepth": 54,
"genotypeQuality": 99,
"alleleDepths": [
25,
29
]
}
],
"variants": [
{
"vid": "1-1043248-C-T",
"chromosome": "chr1",
"begin": 1043248,
"end": 1043248,
"refAllele": "C",
"altAllele": "T",
"variantType": "SNV",
"hgvsG": "NC_000001.11:g.1043248C>T",
"phylopScore": 0.1,
"clinvar": [
{
"id": "RCV000872112.1",
"variationId": 263161,
"reviewStatus": "criteria provided, single submitter",
"alleleOrigins": [
"germline"
],
"refAllele": "C",
"altAllele": "T",
"phenotypes": [
"not provided"
],
"medGenIds": [
"CN517202"
]
},
```

```
"significance": [  
    "likely benign"  
,  
    "lastUpdatedDate": "2019-12-17",  
    "pubMedIds": [  
        "28492532"  
,  
        {"  
            "id": "VCV000263161.2",  
            "reviewStatus": "criteria provided, multiple submitters, no conflicts",  
            "significance": [  
                "likely benign"  
,  
                {"  
                    "refAllele": "C",  
                    "altAllele": "T",  
                    "lastUpdatedDate": "2019-12-17",  
                    "isAlleleSpecific": true  
                }  
,  
                {"  
                    "dbsnp": [  
                        "rs116586548"  
,  
                        {"  
                            "globalAllele": {  
                                "globalMinorAllele": "T",  
                                "globalMinorAlleleFrequency": 0.004393  
                            },  
                            "gnomad": {  
                                "coverage": 38,  
                                "allAf": 0.000681,  
                                "allAn": 264462,  
                                "allAc": 180,  
                                "allHc": 0,  
                                "afrAf": 0.006216,  
                                "afrAn": 23648,  
                                "afrAc": 147,  
                                "afrHc": 0,  
                                "amrAf": 0.000689,  
                                "eurAf": 0.000689,  
                                "easAf": 0.000689,  
                                "othAf": 0.000689  
                            }  
                        }  
                    ]  
                }  
            ]  
        }  
    ]  
]
```

```
"amrAn": 33404,  
"amrAc": 23,  
"amrHc": 0,  
"easAf": 0,  
"easAn": 18830,  
"easAc": 0,  
"easHc": 0,  
"finAf": 0,  
"finAn": 22870,  
"finAc": 0,  
"finHc": 0,  
"nfeAf": 5e-05,  
"nfeAn": 120576,  
"nfeAc": 6,  
"nfeHc": 0,  
"asjAf": 0.000304,  
"asjAn": 9882,  
"asjAc": 3,  
"asjHc": 0,  
"sasAf": 0,  
"sasAn": 28456,  
"sasAc": 0,  
"sasHc": 0,  
"othAf": 0.000147,  
"othAn": 6796,  
"othAc": 1,  
"othHc": 0,  
"maleAf": 0.000564,  
"maleAn": 143614,  
"maleAc": 81,  
"maleHc": 0,  
"femaleAf": 0.000819,  
"femaleAn": 120848,  
"femaleAc": 99,  
"femaleHc": 0,  
"controlsAllAf": 0.000626,  
"controlsAllAn": 113456,  
"controlsAllAc": 71  
,
```

```
"oneKg": {  
    "allAf": 0.004393,  
    "afrAf": 0.016641,  
    "amrAf": 0,  
    "easAf": 0,  
    "eurAf": 0,  
    "sasAf": 0,  
    "allAn": 5008,  
    "afrAn": 1322,  
    "amrAn": 694,  
    "easAn": 1008,  
    "eurAn": 1006,  
    "sasAn": 978,  
    "allAc": 22,  
    "afrAc": 22,  
    "amrAc": 0,  
    "easAc": 0,  
    "eurAc": 0,  
    "sasAc": 0  
},  
"primateAI": [  
{  
    "hgnc": "AGRN",  
    "scorePercentile": 0.12  
}  
,  
{"  
    "revel": {  
        "score": 0.136  
    },  
    "spliceAI": [  
{  
        "hgnc": "AGRN",  
        "acceptorGainScore": 0.1,  
        "acceptorGainDistance": 23,  
        "acceptorLossScore": 0,  
        "acceptorLossDistance": -9,  
        "donorGainScore": 0,  
        "donorGainDistance": -5,  
        "donorLossScore": 0,  
    ]  
}]
```

```
"donorLossDistance": 16
}
],
"topmed": {
"allAf": 0.002055,
"allAn": 125568,
"allAc": 258,
"allHc": 1
},
"transcripts": [
{
"transcript": "ENST00000379370.6",
"source": "Ensembl",
"bioType": "protein_coding",
"codons": "cCg/cTg",
"aminoAcids": "P/L",
"cdnaPos": "1444",
"cdsPos": "1394",
"exons": "8/36",
"proteinPos": "465",
"geneId": "ENSG00000188157",
"hgnc": "AGRN",
"consequence": [
"missense_variant"
],
"hgvsC": "ENST00000379370.6:c.1394C>T",
"hgvsP": "ENSP00000368678.2:p.(Pro465Leu)",
"isCanonical": true,
"polyPhenScore": 0.065,
"polyPhenPrediction": "benign",
"proteinId": "ENSP00000368678.2",
"siftScore": 0.05,
"siftPrediction": "tolerated"
},
{
"transcript": "NM_198576.3",
"source": "RefSeq",
"bioType": "protein_coding",
"codons": "cCg/cTg",
```

```

"aminoAcids": "P/L",
"cdnaPos": "1444",
"cdsPos": "1394",
"exons": "8/36",
"proteinPos": "465",
"geneId": "375790",
"hgnc": "AGRN",
"consequence": [
  "missense_variant"
],
"hgvsC": "NM_198576.3:c.1394C>T",
"hgvsP": "NP_940978.2:p.(Pro465Leu)",
"isCanonical": true,
"polyPhenScore": 0.065,
"polyPhenPrediction": "benign",
"proteinId": "NP_940978.2",
"siftScore": 0.05,
"siftPrediction": "tolerated"
}
]
}
]
}
],

```

Genes

For each gene referenced in the transcripts in the positions section, there is a matching entry in the genes section. The following example shows gene-level annotations from gnomAD, ClinGen Dosage Sensitivity Map, and OMIM.

```

"genes": [
{
  "name": "AGRN",
  "gnomAD": {
    "pLi": 5.47e-07,
    "pRec": 1,
    "pNull": 1.41e-12,
    "synZ": -3.96,

```

```

"misZ": 0.226,
"loef": 0.435
},
"clingenDosageSensitivityMap": {
"haploinsufficiency": "gene associated with autosomal recessive phenotype",
"triplosensitivity": "no evidence to suggest that dosage sensitivity is
associated with clinical phenotype"
},
"omim": [
{
"mimNumber": 103320,
"geneName": "Agrin",
"description": "The AGRN gene encodes agrin, a large and ubiquitous
proteoglycan with multiple isoforms that have diverse functions in different
tissues. Agrin was originally identified as an essential neural regulator that
induces the aggregation of acetylcholine receptors (AChRs) and other
postsynaptic proteins on muscle fibers and is crucial for the formation and
maintenance of the neuromuscular junction (NMJ) (Campanelli et al., 1991;
Burgess et al., 1999; summary by Maselli et al., 2012).",
"phenotypes": [
{
"mimNumber": 615120,
"phenotype": "Myasthenic syndrome, congenital, 8, with pre- and postsynaptic
defects",
"description": "Congenital myasthenic syndromes are genetic disorders of the
neuromuscular junction (NMJ) that are classified by the site of the
transmission defect: presynaptic, synaptic, and postsynaptic. CMS8 is an
autosomal recessive disorder characterized by prominent defects of both the
pre- and postsynaptic regions. Affected individuals have onset of muscle
weakness in early childhood; the severity of the weakness and muscles affected
is variable (summary by Maselli et al., 2012).\n\nFor a discussion of genetic
heterogeneity of CMS, see CMS1A.",
"mapping": "molecular basis of the disorder is known",
"inheritances": [
"Autosomal recessive"
]
}
]
}
]
}

```

```
]
}
]
}
```

DRAGEN ORA Compression and Decompression

DRAGEN ORA Compression is a fully lossless compression, that compresses `*.fastq` and `*.fastq.gz` files into `*.fastq.ora` files. DRAGEN ORA supports FASTQ generated by Illumina sequencing systems. When using the ORA format, the md5 checksum of the FASTQ content is preserved after a compression and decompression cycle to ensure a lossless compression.

DRAGEN ORA Compression requires a separate license. Decompression and ingestion of `*.fastq.ora` files into the DRAGEN map/align does not require a license. If the DRAGEN server is connected to a network, DRAGEN ORA Compression can be used after installing DRAGEN v3.8 or later. If your DRAGEN server is offline, contact Illumina Customer Service.

For human data generated by the NovaSeq 6000, NextSeq 1000, or NextSeq 2000 sequencing systems, the compression ratio is expected to be up to 6x compared to the `*.fastq.gz`. The compressed file uses the `*.fastq.ora` extension.

Input of DRAGEN ORA Compression is `*.fastq` or `*.fastq.gz`. The input can be a single file or a list of files. A list of files can be specified on the command line, or from a `*.fastq-list.csv` generated by the BCL Convert BaseSpace Sequence Hub App or DRAGEN BCL Convert. Input located in local storage, AWS S3 or Azure Blob store is supported.

`*.fastq.ora` files are decompressed into `*.fastq.gz`.

 `*.fastq.ora` can be generated starting from BCL. To convert BCL into `*.fastq.ora` specific commands need to be used. Follow the [DRAGEN ORA Compression from BCL](#) instructions.

ORA Reference

To compress or decompress ORA files, an ORA reference file must be supplied and a reference directory specified. An ORA reference file can be downloaded from the [Illumina DRAGEN Bio-IT Platform Product Files](#) page.

Use the following steps to specify an ORA reference directory.

1. Download the `oradata-2.tar.gz` from the [Illumina DRAGEN Bio-IT Platform Product Files](#) page.
2. Move the file to the directory where the files should be extracted, and enter the following command: `tar -xzvf oradata-2.tar.gz`.
3. Set the `--ora-reference` command line option to the extracted `/oradata` folder path.

Command Line Options

The following example command contains the required DRAGEN ORA compression options.

```
dragen --enable-map-align false --ora-input <FILE> --enable-ora true --ora-reference <...> --output-directory <...>
```

or

```
dragen --enable-map-align false --fastq-list <FILE .csv> --enable-ora true --ora-reference <...> --output-directory <...>
```

The following example command contains the required ORA decompression options.

```
dragen --enable-map-align false --ora-input <FILE> --enable-ora true --ora-decompress true --ora-reference <...> --output-directory <...>
```

The following examples command contains the required options to compress FASTQs of a fastq-list.csv file containing multiple samples.

When all samples must be compressed:

```
dragen --enable-map-align false --fastq-list <FILE .csv> --enable-ora true --fastq-list-all-samples true --ora-reference <...> --output-directory <...>
```

When only specific samples must be compressed:

```
dragen --enable-map-align false --fastq-list <FILE .csv> --enable-ora true --fastq-list-sample-id <sample> --ora-reference <...> --output-directory <...>
```

The following examples command contains the required options to achieve an interleaved compression of paired read files from a fastq-list.csv file :

```
dragen --enable-map-align false --fastq-list <FILE .csv> --enable-ora true --ora-interleaved-compression true --ora-reference <...> --output-directory <...>
```

The following example command prints the file information summary of an ORA compressed file.

Compression or decompression is not performed.

```
dragen --enable-map-align false --ora-input <FILE> --enable-ora=true --ora-print-file-info
```

The following example command compares FASTQ file checksum and decompressed FASTQ.ORA file checksum and outputs *ORA integrity check successful* if both checksums are equal or *integrity check failed* if checksums are not equal.

```
dragen --enable-map-align false --ora-input <FILE> --enable-ora=true --ora-reference <...> --ora-check-file-integrity=true`
```

The following are the command line options for running DRAGEN ORA Compression and Decompression.

Option	Required	Description
--enable-map-align	Yes	Set to <code>false</code> to perform compression only. Only the compression license gets deducted.
		Set to <code>true</code> to perform the compression in parallel of the map/align step. Both the compression license and DRAGEN license are deducted. When set to true all the options required to process with the map/align step must be provided.
--enable-ora	Yes	Set to <code>true</code> to enable FASTQ file compression and decompression. Decompression must be enabled using the <code>--ora-decompress</code> option.
--ora-reference	Yes	Path to the directory that contains the compression reference and index file.
--ora-input	Yes (or --fastq-list)	Specifies the input files for compression or decompression.
--fastq-list	Yes (or --ora-input)	Specifies a <code>.csv</code> file with list of FASTQ files to be compressed. The option is not specific to the DRAGEN ORA Compression and the usage is explained in the FASTQ CSV File Format Section of this manual.
--ora-input2	No	Used for interleaved compression of paired read files when input files are specified with <code>--ora-input</code> . Specify the paired read files corresponding to files specified in <code>--ora-input</code> to achieve paired read file compression into one single interleaved file. The number of files and the order of paired read files in <code>--ora-input</code> and <code>--ora-input2</code> should match.

Option	Required	Description
--ora-interleaved-compression	No	Used for interleaved compression of paired read files when input files are specified with --fastq-list. Set to true to enable paired read file compression into one single interleaved file. Each line of the fastq-list.csv file is the two corresponding paired read files with same count of reads.
--ora-decompress	No	Set to true to enable decompress mode. The default value is false.
--force	No	Compresses to output directory even if the compressed file already exists. The existing compressed file is overwritten.
--ora-threads-per-file <#>-	No	Manually controls the number of CPU threads for compressing each FASTQ input file. The default value is 8.
--ora-parallel-files <#>	No	Manually controls the number of input FASTQ files processed in parallel. The default value is 4.
--ora-use-hw	No	Set to true to enable hardware acceleration or to false to disable hardware acceleration. The default value is true. When set to false on on-site systems, DRAGEN ORA Compression and Decompression can be launched in parallel to other processes that use FPGA. This execution of DRAGEN ORA Compression or Decompression, in parallel with other DRAGEN processes, is not supported on the cloud.
--ora-print-file-info	No	Prints file information summary of ORA compressed files. This option cannot be used simultaneously with the ora-decompress and --ora-check-file-integrity options.

Option	Required	Description
--ora-check-file-integrity	No	Set to <code>true</code> to perform and output result of FASTQ file and decompressed FASTQ.ORA integrity check. The default value is <code>false</code> . This option cannot be performed in the same command line as the compression itself since it requires <code>fastq.ora</code> format for the <code>--ora-input</code> argument.
--ora-enable-md5	No	Set to <code>true</code> to compute md5 checksum of <code>fastq.ora</code> files during the compression and generate an <code>ora.md5sum</code> file with md5 checksum printed.
--ora-delete-input-files	No	Set to <code>true</code> to automatically delete the input FASTQ file from the disk upon completion of compression
--ora-original-name	No	At decompression, set to <code>true</code> to retrieve the name of the original FASTQ before compression. Default re-uses the name of the FASTQ.ORA provided as input.

Use the `--output-directory` option to specify the directory to store output compressed/decompressed files.

Interleaved Compression

There are two methods to achieve a paired compression aka interleaved compression:

- when using `--ora-input` and `--ora-input2`. The nth file of the `--ora-input` list is compressed together with the nth file of the `--ora-input2`
- when using `--fastq-list` and `--ora-interleaved-compression` set to `true`. The paired read files from the nth line of `fastq-list.csv` are compressed together

Both files are interleaved within a single ORA output file with file name containing `-interleaved`. Using these options to compress paired files together improves compression by up to 10%. If decompressing an ORA file that contains paired data, the file is automatically decompressed to two separate files. To map an ORA file that contains paired interleaved data with the DRAGEN mapper, use the `--interleaved` option.

How to use ORA input files with DRAGEN Map/Align

DRAGEN can directly process ORA files. The same options as the other FASTQ input file types can be used. To use the ORA file, replace the FASTQ file name with the ORA file name and specify the ORA reference directory using `--ora-reference`.

The following command represents paired-end in two matched ORA FASTQ files (-1 and -2 options).

```
dragen -r <REF_DIR> -1 <fastq.ora1> -2 <fastq.ora2> \
--ora-reference <ORADATA_DIR> \
--output-directory <OUT_DIR> --output-file-prefix <OUT_PREFIX> \
--RGID <RGID> --RGSM <RGSM>
```

Hardware-Accelerated Compression and Decompression

Gzip compression is ubiquitous in bioinformatics. FASTQ files are often gzipped, and the BAM format itself is a specialized version of gzip. For that reason, the DRAGEN BioIT processor provides hardware support for accelerating compression and decompression of gzipped data. If your input files are gzipped, DRAGEN detects that and decompresses the files automatically. If your output is BAM files, then the files are automatically compressed.

DRAGEN provides standalone command-line utilities to enable you to compress or decompress arbitrary files. These utilities are analogous to the Linux gzip and gunzip commands, but are named *dzip* and *dunzip* (dragen zip and dragen unzip). Both utilities are able to accept as input a single file, and produce a single output file with the .gz file extension removed or added, as appropriate. For example:

```
dzip file1      # produces output file file1.gz
dunzip file2.gz # produces output file file2
```

Currently, *dzip* and *dunzip* have the following limitations and differences from gzip/gunzip:

- Each invocation of these tools can handle only a single file. Additional file names (including those produced by a wildcard * character) are ignored.
- They cannot be run at the same time as the DRAGEN host software.
- They do not support the command line options found in gzip and gunzip (eg, `--recursive`, `--fast`, `--best`, `--stdout`).

Usage Reporting

As part of the install process, a daemon (`dragen_ljcd`) is created (or stopped then restarted). This background process self-activates at the end of each day to upload DRAGEN host software usage to an Illumina server. Information includes date, duration, size (number of bases), status of each run, and

software version used.

Communication to the illumina server is secured by encryption. If there is a communication error, the daemon retries until the next morning. If the upload continues to fail, communication is tried again the following night until communication succeeds. This means that during working hours, the system resources remain fully available and are not in any way hampered by this background activity.

To check the current license usage, use the *dragen_lic* command.

To generate a usage report, your server must meet the following requirements:

- 256 GB RAM and 2 TB HDD for 300x germline single sample coverage.
- T/N analysis coverage.
- 6 TB and 512 GB RAM.

The usage report does not contain the following information:

- Number of joint genotyper input files.
- GATK gVCF input to gVCF genotyper.
- Mixing gVCFs from different callers, such as joint calling and gVCF genotyper.

Troubleshooting

If the DRAGEN system does not seem to be responding, do the following:

1. To determine if the DRAGEN system is hanging, follow the instructions in [How to Determine if the System is Hanging on page 675](#).
2. Collect diagnostic information after a hang, or a crash, as described in [Sending Diagnostic Information to Illumina Support on page 675](#).
3. After all information has been collected, reset your system, if needed, as described in [Resetting Your System after a Crash or Hang on page 675](#).

How to Determine if the System is Hanging

The DRAGEN system has a watchdog to monitor the system for hangs. If a run seems to be taking longer than it should, the watchdog may not be detecting the hang. Here are some things to try:

- Run the *top* command to find the active DRAGEN process. If your run is healthy, you should expect to see it consuming over 100% of the CPU. If it is consuming 100% or less, then your system may be hanging.
- Run the *du -s* command in the directory of the output BAM/SAM file. During a normal run, this directory should be growing with either intermediate output data (when sort is enabled) or BAM/SAM data.

Sending Diagnostic Information to Illumina Support

Illumina would like your feedback on your DRAGEN system, including any reports of system malfunction. In the event of a crash, hang, or watchdog fault, run the *sosreport* command to collect diagnostic and configuration information, as follows:

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

This command takes several minutes to execute and reports the location where it has saved the diagnostic information in */staging/tmp*. Include the report when you submit a ticket for Illumina Technical Support.

Resetting Your System after a Crash or Hang

If the DRAGEN system crashes or hangs, the *dragen_reset* utility must be run to reinitialize the hardware and software. This utility is automatically executed by the host software any time it detects an unexpected condition. In this case, the host software shows the following message:

```
Running dragen_reset to reset DRAGEN Bio-IT processor and software
```

If the software is hanging, please collect diagnostic information as described in subsection [Sending Diagnostic Information to Illumina Support](#) on page 675 and then execute *dragen_reset* manually, as follows:

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

Any execution of *dragen_reset* requires the reference genome to be reloaded to the DRAGEN board. The host software automatically reloads the reference on the next execution.

Command-Line Options

This section provides information on all the DRAGEN command-line options, including the name used in the configuration file, the command-line equivalent, a description, and the range of values.

! After upgrading to a new version of DRAGEN, it is recommended to first run with the default DRAGEN options, including all filtering options, and then add any specific filters only if needed.

General Software Options

The following options are in the default section of the configuration file. The default section is at the top of the configuration file and does not have a section name (eg, [Aligner]) associated with it. Some mandatory fields must be specified on the command line and are not present in configuration files.

Name	Description	Command-Line Equivalent	Value
append-read-index-to-name	By default, DRAGEN names both mate ends of pairs the same. When set to true, DRAGEN appends /1 and /2 to the two ends.	--append-read-index-to-name	<ul style="list-style-type: none"> • true • false
aws-s3-region	Specifies the geographical region of AWS S3 buckets.	--aws-3-region	
bam-input	Specifies aligned BAM file for input to the DRAGEN variant caller.	-b, --bam-input	
bcl-conversion-only	Performs Illumina BCL conversion to FASTQ format.	--bcl-conversion-only	<ul style="list-style-type: none"> • true • false
bcl-input-directory	Inputs BCL directory for BCL conversion.	--bcl-input-directory	
bcl-only-lane	For BCL input, the option converts only specified lane number. By default, all lanes are converted.	--bcl-only-lane	1-8

Name	Description	Command-Line Equivalent	Value
sample-sheet	For BCL input, the option sets the path to <code>SampleSheet.csv</code> file. The default location is the BCL root directory.	--sample-sheet	
strict-mode	For BCL input, the option cancels analysis if any files are missing. The default value is false by default.	--strict-mode	<ul style="list-style-type: none"> • true • false
first-tile-only	Converts only the first tile of each lane during BCL conversion. Use for testing or debugging.	--first-tile-only	<ul style="list-style-type: none"> • true • false
run-info	Sets the path to <code>RunInfo.xml</code> file. The default is <code><flow cell>/RunInfo.xml</code> .	--run-info	
bcl-sampleproject-subdirectories	For BCL conversion, the option outputs to subdirectories based on sample sheet <code>Sample_Project</code> column.	--bcl-sampleproject-subdirectories	
no-lane-splitting	Disables splitting output FASTQ files by lane. The default value is false.	--no-lane-splitting	<ul style="list-style-type: none"> • true • false
bcl-only-matched-reads	Specifies if unmapped reads are output to files marked as Undetermined. The default value is false.	bcl-only-matched-reads	<ul style="list-style-type: none"> • true • false
bcl-use-hw	If set to false, the option prevents DRAGEN FPGA acceleration during BCL conversion. The default value is true.	--bcl-use-hw	<ul style="list-style-type: none"> • true • false

Name	Description	Command-Line Equivalent	Value
bcl-num-parallel-tiles	Specifies the number of tiles to process in parallel. The default value is dynamically determined.	--bcl-num-parallel-tiles	<ul style="list-style-type: none"> • 1-<nproc>
bcl-num-conversion-threads	Specifies the number of conversion threads per tile. The default value is dynamically determined.	--bcl-num-conversion-threads	<ul style="list-style-type: none"> • 1-<nproc>
bcl-num-compression-threads	Specifies the number of CPU threads for output fastq.gz compression. The default value is dynamically determined.	--bcl-num-compression-threads	<ul style="list-style-type: none"> • 1-<nproc>
bcl-num-decompression-threads	Specifies the number of CPU threads for BCL input decompression. The default value is dynamically determined.	--bcl-num-decompression-threads	<ul style="list-style-type: none"> • 1-<nproc>
shared-thread-odirect-output	Uses alternative shared-thread ODIRECT file output. The default value is false.	--shared-thread-odirect-output	<ul style="list-style-type: none"> • true • false
build-hash-table	Generates a reference hash table.	--build-hash-table	<ul style="list-style-type: none"> • true • false
cram-input	Specifies the CRAM file input for the DRAGEN variant caller.	--cram-input	
dbsnp	Sets the path to the variant annotation database VCF (or *.vcf.gz) file.	--dbsnp	
enable-auto-multifile	Imports subsequent segments of the *_001. {dbam, fastq} files.	--enable-auto-multifile	<ul style="list-style-type: none"> • true • false
enable-bam-indexing	Enables generation of a BAI index file.	--enable-bam-indexing	<ul style="list-style-type: none"> • true • false

Name	Description	Command-Line Equivalent	Value
enable-cram-indexing	Enables generation of a CRAI index file.	--enable-cram-indexing	<ul style="list-style-type: none"> • true • false
enable-cnv	Enables copy number variant (CNV).	--enable-cnv	<ul style="list-style-type: none"> • true • false
enable-duplicate-marking	Enables the flagging of duplicate output alignment records.	--enable-duplicate-marking	<ul style="list-style-type: none"> • true • false
enable-map-align-output	Enables saving the output from the map/align stage. If only running map/align, the default value is true. If running the variant caller, the default value is false.	--enable-map-align-output	<ul style="list-style-type: none"> • true • false
enable-methylation-calling	Automatically adds tags related to methylation and outputs a single BAM for methylation protocols.	--enable-methylation-calling	<ul style="list-style-type: none"> • true • false
enable-sampling	Automatically detects paired-end parameters by running a sample through the mapper/aligner.	--enable-sampling	<ul style="list-style-type: none"> • true • false
enable-sort	Enables sorting after mapping/alignment.	--enable-sort	<ul style="list-style-type: none"> • true • false
enable-variant-caller	Enables the variant caller. (default=false)	--enable-variant-caller	<ul style="list-style-type: none"> • true • false
enable-variant-deduplication	Enables variant deduplication. The default value is false.	--enable-variant-deduplication	<ul style="list-style-type: none"> • true • false
enable-vcf-compression	Enables compression of VCF output files. The default value is true.	--enable-vcf-compression	<ul style="list-style-type: none"> • true • false

Name	Description	Command-Line Equivalent	Value
enable-vcf-indexing	Outputs a *.tbi index file in addition to the output VCF/gVCF. The default is true.	--enable-vcf-indexing	<ul style="list-style-type: none"> • true • false
fastq-file1	Specifies FASTQ file to input to the DRAGEN pipeline. Gzipped format can be used.	-1, --fastq-file1	
fastq-file2	Specifies second FASTQ file with paired-end reads to input.	-2, --fastq-file2	
fastq-list	Specifies CSV file that contains a list of FASTQ files to process.	--fastq-list	
fastq-list-all-samples	Disables processing of all samples together, regardless of the RGSM value.	--fastq-list-all-samples	<ul style="list-style-type: none"> • true • false
fastq-n-quality	Specifies the base call quality to output for N bases. Automatically added to fastq-n-quality for all output N bases.	--fastq-n-quality	<ul style="list-style-type: none"> • 0–255
fastq-offset	Sets the FASTQ quality offset value.	--fastq-offset	<ul style="list-style-type: none"> • 33 • 64
filter-flags-from-output	Filters output alignments with any bits set in val present in the flags field. Hex and decimal values accepted.	--filter-flags-from-output	
force	Forces overwrite of existing output file.	-f	

Name	Description	Command-Line Equivalent	Value
force-load-reference	Forces loading of the reference and hash tables before starting the DRAGEN pipeline.	-l	
generate-md-tags	Generates MD tags with alignment output records. The default value is false.	--generate-md-tags	<ul style="list-style-type: none"> • true • false
generate-sa-tags	Generates SA:Z tags for records that have chimeric or supplemental alignments.	--generate-sa-tags	<ul style="list-style-type: none"> • true • false
generate-zs-tags	Generate ZS tags for alignment output records. The default value is false.	--generate-zs-tags	<ul style="list-style-type: none"> • true • false
ht-alt-liftover	SAM format liftover file of alternate contigs in reference.	--ht-alt-liftover	
ht-mask-bed	Specifies the BED file for base masking.	--ht-mask-bed	
ht-allow-mask-and-liftover	Allows the hash table builder to run with both ht-alt-liftover and ht-mask-bed. Default is false.	--ht-allow-mask-and-liftover	<ul style="list-style-type: none"> • true • false
ht-build-rna-hashtable	Enables generation of RNA hash table. The default value is false.	--ht-build-rna-hashtable	<ul style="list-style-type: none"> • true • false
ht-cost-coeff-seed-freq	Sets cost coefficient of extended seed frequency.	--ht-cost-coeff-seed-freq	
ht-cost-coeff-seed-len	Sets cost coefficient of extended seed length.	--ht-cost-coeff-seed-len	
ht-cost-penalty-incr	Sets cost penalty to incrementally extend a seed another step.	--ht-cost-penalty-incr	

Name	Description	Command-Line Equivalent	Value
ht-cost-penalty	Sets cost penalty to extend a seed by any number of bases.	--ht-cost-penalty	
ht-decoys	Specifies the path to a decoys file.	--ht-decoys	
ht-max-dec-factor	Sets the maximum decimation factor for seed thinning.	--ht-max-dec-factor	
ht-max-ext-incr	Sets the maximum bases to extend a seed by in one step.	--ht-max-ext-incr	
ht-max-ext-seed-len	Specifies the maximum extended seed length.	--ht-max-ext-seed-len	
ht-max-seed-freq	Sets the maximum allowed frequency for a seed match after extension attempts.	--ht-max-seed-freq	• 1–256
ht-max-table-chunks	Specifies the maximum ~1 GB thread table chunks in memory at one time.	--ht-max-table-chunks	
ht-mem-limit	Specifies the memory limit (hash table + reference) in units (kb, MB, GB).	--ht-mem-limit	
ht-methylated	Automatically generates C->T and G->A converted reference hash tables.	--ht-methylated	• true • false
ht-num-threads	Sets maximum worker CPU threads for building hash table.	--ht-num-threads	
ht-rand-hit-extend	Includes a random hit with each EXTEND record of the frequency record.	--ht-rand-hit-extend	
ht-rand-hit-hifreq	Includes a random hit with each HIFREQ record.	--ht-rand-hit-hifreq	

Name	Description	Command-Line Equivalent	Value
ht-ref-seed-interval	Specifies the number of positions per reference seed.	--ht-ref-seed-interval	
ht-reference	References file in FASTA format to build a hash table.	--ht-reference	
ht-seed-len	Sets initial seed length to store in hash table.	--ht-seed-len	
ht-size	Specifies the size of hash table in units (kb, MB, GB).	--ht-size	
ht-soft-seed-freq-cap	Specifies the soft seed frequency cap for thinning.	--ht-soft-seed-freq-cap	
ht-suppress-decoys	Suppresses the use of a decoys file when building a hash table.	--ht-suppress-decoys	
ht-target-seed-freq	Sets the target seed frequency for seed extension.	--ht-target-seed-freq	
input-qname-suffix-delimiter	Controls the delimiter used for append-read-index-to-name and for detecting matching pair names with BAM input.	--input-qname-suffix-delimiter	• / • . • :
interleaved	Specifies the interleaved paired-end reads in single FASTQ.	-i	
intermediate-results-dir	Specifies directory to store intermediate results in (eg, sort partitions).	--intermediate-results-dir	
lic-no-print	Suppresses the license status message at the end of a run.	--lic-no-print	• true • false

Name	Description	Command-Line Equivalent	Value
lic-server	Specifies the license server for cloud sites: <code>http://<base64_use>:<base64_password>@<path to server></code>	--lic-server	
lic-instance-id-location	Use this option to override the default cloud instance ID location	--lic-instance-id-location	
lic-credentials	Path to license credentials file that specifies the license credentials and domain.	--lic-credentials	
methylation-generate-cytosine-report	Generates a genome-wide cytosine methylation report.	--methylation-generate-cytosine-report	<ul style="list-style-type: none"> • true • false
methylation-generate-mbias-report	Generates a per system cycle methylation bias report.	--methylation-generate-mbias-report	<ul style="list-style-type: none"> • true • false
methylation-TAPS	If input assays are generated by TAPS, the option is set to true.	--methylation-TAPS	<ul style="list-style-type: none"> • true • false
methylation-match-bismark	If true, the option matches Bismark tags exactly, including bugs.	--methylation-match-bismark	<ul style="list-style-type: none"> • true • false
methylation-protocol	Describes library protocol for methylation analysis.	--methylation-protocol	<ul style="list-style-type: none"> • none • directional • nondirectional • directional-complement
num-threads	Specifies the number of processor threads to use.	-n, --num-threads	
output-directory	Specifies the output directory.	--output-directory	

Name	Description	Command-Line Equivalent	Value
output-file-prefix	Outputs file name prefix to use for all files generated by the pipeline.	--output-file-prefix	
output-format	Sets the format of the output file from the map/align stage. The following values are valid <ul style="list-style-type: none">• BAM (default)• CRAM (lossless)• SAM• DBAM (proprietary binary format)	--output-format	<ul style="list-style-type: none">• BAM• CRAM• SAM• DBAM
pair-by-name	Shuffles the order of BAM input records so paired-end mates are processed together.	--pair-by-name	
pair-suffix-delimiter	Changes the delimiter character for suffixes.	--pair-suffix-delimiter	<ul style="list-style-type: none">• /• .• :
preserve-bqsr-tags	Determines whether to preserve BI and BD flags from the input BAM file, which can cause problems with hard clipping.	--preserve-bqsr-tags	<ul style="list-style-type: none">• true• false
preserve-map-align-order	Produces output file that preserves original order of reads in the input file.	--preserve-map-align-order	<ul style="list-style-type: none">• true• false
qc-coverage-region-1	Generates coverage region report using bed file 1.	--qc-coverage-region-1	
qc-coverage-region-2	Generates coverage region report using bed file 2.	--qc-coverage-region-2	
qc-coverage-region-3	Generates coverage region report using bed file 3.	--qc-coverage-region-3	
qc-coverage-reports-1	Describes the types of reports requested for qc-coverage-region-1.	--qc-coverage-reports-1	<ul style="list-style-type: none">• full_res• cov_report

Name	Description	Command-Line Equivalent	Value
qc-coverage-reports-2	Describes the types of reports requested for qc-coverage-region-2.	--qc-coverage-reports-2	<ul style="list-style-type: none"> • full_res • cov_report
qc-coverage-reports-3	Describes the types of reports requested for qc-coverage-region-3.	--qc-coverage-reports-3	<ul style="list-style-type: none"> • full_res • cov_report
qc-coverage-region-1-thresholds	Declares the thresholds to use in cov_report for qc-coverage-region-1.	--qc-coverage-region-1-thresholds	List of up to 11 numbers separated by commas
qc-coverage-region-2-thresholds	Declares the thresholds to use in cov_report for qc-coverage-region-2.	--qc-coverage-region-2-thresholds	List of up to 11 numbers separated by commas
qc-coverage-region-3-thresholds	Declares the thresholds to use in cov_report for qc-coverage-region-3.	--qc-coverage-reports-3	List of up to 11 numbers separated by commas
ref-dir	Specifies the directory containing the reference hash table. If the reference is not already loaded into the DRAGEN card, the option automatically loads the reference.	-r, --ref-dir	
ref-sequence-filter	Outputs only reads mapping to the reference sequence.	--ref-sequence-filter	
remove-duplicates	If true, the option removes duplicate alignment records instead of only flagging them.		<ul style="list-style-type: none"> • true • false
RGCN	Specifies the read group sequencing center name.	--RGCN	

Name	Description	Command-Line Equivalent	Value
RGCN-tumor	Specifies the read group sequencing center name for tumor input.	--RGCN-tumor	
RGDS	Provides the read group description.	--RGDS	
RGDS-tumor	Provides the read group description for tumor input.	--RGDS-tumor	
RGDT	Specifies the read group run date.	--RGDT	
RGDT-tumor	Specifies the read group run date for tumor input.	--RGDT-tumor	
RGID	Specifies read group ID.	--RGID	
RGID-tumor	Specifies read group ID for tumor input.	--RGID-tumor	
RGLB	Specifies the read group library.	--RGLB	
RGLB-tumor	Specifies the read group library for tumor input.	--RGLB-tumor	
RGPI	Specifies the read group predicted insert size.	--RGPI	
RGPI-tumor	Specifies the read group predicted insert size for tumor input.	--RGPI-tumor	
RGPL	Specifies the read group sequencing technology.	--RGPL	
RGPL-tumor	Specifies the read group sequencing technology for tumor input.	--RGPL-tumor	
RGPU	Specifies the read group platform unit.	--RGPU	
RGPU-tumor	Specifies read group platform unit for tumor input.	--RGPU-tumor	

Name	Description	Command-Line Equivalent	Value
RGSM	Specifies read group sample name.	--RGSM	
RGSM-tumor	Specifies read group sample name for tumor input.	--RGSM-tumor	
sample-size	Specifies number of reads to sample when enable-sampling is true.	--sample-size	
sample-sex	Specifies the sex of the sample.	--sample-sex	
strip-input-qname-suffixes	Determines whether to strip read-index suffixes (eg, /1 and /2) from input QNAMEs. If set to false, the option preserves entire name.	--strip-input-qname-suffixes	<ul style="list-style-type: none"> • true • false
tumor-bam-input	Specifies aligned BAM file for the DRAGEN variant caller in somatic mode.	--tumor-bam-input	
tumor-cram-input	Specifies aligned CRAM file for the DRAGEN variant caller in somatic mode.	--tumor-cram-input	
tumor-fastq-list	Inputs a CSV file containing a list of FASTQ files for the mapper, aligner, and somatic variant caller.	--tumor-fastq-list	
tumor-fastq-list-sample-id	Specifies the sample ID for the list of FASTQ files specified by tumor-fastq-list.	--tumor-fastq-list-sample-id	
tumor-fastq1	Inputs FASTQ file for the DRAGEN pipeline using the variant caller in somatic mode. The input file can be gzipped.	--tumor-fastq1	

Name	Description	Command-Line Equivalent	Value
tumor-fastq2	Inputs second FASTQ file. Reads are paired to tumor-fastq1 reads for the DRAGEN pipeline using the variant caller in somatic mode. The input file can be gzipped.	--tumor-fastq2	
vd-eh-vcf	Inputs the ExpansionHunter VCF file for variant deduplication. The input file can be gzipped.	--vd-eh-vcf	
vd-output-match-log	Outputs a file that describes the variants that matched during deduplication. The default value is false.	--vd-output-match-log	<ul style="list-style-type: none"> • true • false
vd-small-variant-vcf	Inputs small variant VCF file for variant deduplication. The input file can be gzipped.	--vd-small-variant-vcf	
vd-sv-vcf	Inputs structural variant VCF for variant deduplication. The input file can be gzipped.	--vd-sv-vcf	
verbose	Enables verbose output from DRAGEN.	-v	
version	Prints the version and exits.	-V	

Mapper Options

The following options are in the [Mapper] section of the configuration file. For more detailed information on the following options, see [DNA Mapping on page 79](#).

Name	Description	Command-Line Equivalent	Value
ann-sj-max-indel	Specifies maximum indel length to expect near an annotated splice junction.	--Mapper.ann-sj-max-indel	<ul style="list-style-type: none"> • 0–63
edit-chain-limit	For edit-mode 1 or 2, the option sets maximum seed chain length in a read to qualify for seed editing.	--Mapper.edit-chain-limit	<ul style="list-style-type: none"> • edit-chain-limit >= 0
edit-mode	Controls when seed editing is used. The following values represent the different edit modes: <ul style="list-style-type: none"> • 0 is no edits • 1 is chain length test • 2 is paired chain length test • 3 is full seed edits 	--Mapper.edit-mode	<ul style="list-style-type: none"> • 0–3
edit-read-len	For edit-mode 1 or 2, controls the read length for edit-seed-num seed editing positions.	--Mapper.edit-read-len	<ul style="list-style-type: none"> • edit-read-len > 0
edit-seed-num	For edit-mode 1 or 2, controls the requested number of seeds per read to allow editing on.	--Mapper.edit-seed-num	<ul style="list-style-type: none"> • edit-seed-num >= 0
enable-map-align	Enable the mapper/aligner (Default=true)	--enable-map-align	<ul style="list-style-type: none"> • true • false
map-orientations	Restricts the orientation of read mapping to only forward in the reference genome or only reverse-complemented. The following values represent the different orientations (paired end requires normal): <ul style="list-style-type: none"> • 0 is normal (paired-end inputs must use normal) • 1 is reverse-complemented • 2 is no forward 	--Mapper.map-orientations	<ul style="list-style-type: none"> • 0–2
max-intron-bases	Specifies maximum intron length reported.	--Mapper.max-intron-bases	

Name	Description	Command-Line Equivalent	Value
min-intron-bases	Specifies minimum reference deletion length reported as an intron.	--Mapper.min-intron-bases	
seed-density	Controls requested density of seeds from reads queried in the hash table	--Mapper.seed-density	<ul style="list-style-type: none"> • 0 > seed-density > 1

Aligner Options

The following options are in the [Aligner] section of the configuration file. For more information, see [DNA Aligning on page 82](#)

Name	Description	Command-Line Equivalent	Value
aln-min-score	A signed integer that specifies a minimum acceptable alignment score to report the baseline for MAPQ. When using local alignments (global is 0), aln-min-score is computed by the host software as 22 * match-score. When using global alignments (global is 1), aln-min-score is set to -1000000. The DRAGEN computation can be overridden by setting aln-min-score in configuration file.	--Aligner.aln-min-score	<ul style="list-style-type: none"> • -2,147,483,648 to 2,147,483,647

Name	Description	Command-Line Equivalent	Value
clip-pe-overhang	When set to a nonzero value, clips 3 read ends overhanging their mate's 5 ends as aligned. The following values represent the different options: <ul style="list-style-type: none"> • 1 is soft-clip overhang • 2 is hard-clip 	--Aligner.clip-pe-overhang	• 0–2
dedup-min-qual	Specifies a minimum base quality for calculating read quality metric for deduplication.	--Aligner.dedup-min-qual	• 0–63
en-alt-hap-aln	Allows haplotype alignments to be output as supplementary.	--Aligner.en-alt-hap-aln	• 0–1
en-chimeric-aln	Allows chimeric alignments to be output as supplementary.	--Aligner.en-chimeric-aln	• 0–1
gap-ext-pen	Specifies the penalty for extending a gap.	--Aligner.gap-ext-pen	• 0–15
gap-open-pen	Specifies the penalty for opening a gap (ie, insertion or deletion).	gap-open-pen	• 0–127
global	Controls whether alignment is end-to-end in the read. The following values represent the different alignments: <ul style="list-style-type: none"> • 0 is local alignment (Smith-Waterman) • 1 is global alignment (Needleman-Wunsch) 	--Aligner.global	• 0–1

Name	Description	Command-Line Equivalent	Value
hard-clips	Specifies alignments for hard clipping. The following values represent the different alignments: <ul style="list-style-type: none">• Bit 0 is primary• Bit 1 is supplementary• Bit 2 is secondary	--Aligner.hard-clips	<ul style="list-style-type: none">• 3 bits
map-orientations	Constrains orientations to accept forward-only, reverse-complement only, or any alignments. The following values represent the different orientations: <ul style="list-style-type: none">• 0 is any• 1 is forward only• 2 is reverse only	--Aligner.map-orientations	<ul style="list-style-type: none">• 0–2
mapq-max	Specifies ceiling on reported MAPQ. The default value is 60.	--Aligner.mapq-max	<ul style="list-style-type: none">• 0–255
mapq-strict-js	Specific to RNA. When set to 0, a higher MAPQ value is returned, expressing confidence that the alignment is at least partially correct. When set to 1, a lower MAPQ value is returned, expressing the splice junction ambiguity.	--mapq-strict-js	<ul style="list-style-type: none">• 0–1
match-n-score	A signed integer that specifies the score increment for matching where a read or reference base is N.	--Aligner.match-n-score	<ul style="list-style-type: none">• -16–15

Name	Description	Command-Line Equivalent	Value
match-score	Specifies the score increment for matching reference nucleotide.	--Aligner.match-score	<ul style="list-style-type: none"> • When global = 0, match-score > 0 • When global = 1, match-score >= 0
max-rescues	Specifies maximum rescue alignments per read pair. The default value is 10.	--max-rescues	• 0–1023
min-score-coeff	Sets adjustment to aln-min-score per read base.	--Aligner.min-score-coeff	• -64–63.999
mismatch-pen	Defines the score penalty for a mismatch.	--Aligner.mismatch-pen	• 0–63
no-unclip-score	When set to 1, the option removes any unclipped bonus (unclip-score) contributing to an alignment from the alignment score before further processing.	--Aligner.no-unclip-score	• 0–1
no-unpaired	Determines if only properly paired alignments should be reported for paired reads.	--Aligner.no-unpaired	• 0–1
pe-max-penalty	Specifies the maximum pairing score penalty for unpaired or distant ends.	--Aligner.pe-max-penalty	• 0–255

Name	Description	Command-Line Equivalent	Value
pe-orientation	<p>Specifies the expected paired-end orientation.</p> <p>The following values represent the different orientations:</p> <ul style="list-style-type: none"> • 0 is FR (default) • 1 is RF • 2 is FF 	--Aligner.pe-orientation	<ul style="list-style-type: none"> • 0–2
rescue-sigmas	<p>Sets deviations from the mean read length used for rescue scan radius.</p> <p>The default value is 2.5.</p>	--Aligner.rescue-sigmas	
sec-aligns	Restricts the maximum number of secondary (suboptimal) alignments to report per read.	--Aligner.sec-aligns	<ul style="list-style-type: none"> • 0–4095
sec-aligns-hard	If set to 1, forces the read to be unmapped when not all secondary alignments be output.	--Aligner.sec-aligns-hard	<ul style="list-style-type: none"> • 0–1
sec-phred-delta	Controls which secondary alignments are emitted. Only secondary alignments within this Phred value of the primary are reported.	--Aligner.sec-phred-delta	<ul style="list-style-type: none"> • 0–255
sec-score-delta	Determines the pair score threshold below primary that secondary alignments are allowed.	--Aligner.sec-score-delta	
supp-aligns	Restricts the maximum number of supplementary (chimeric) alignments to report per read.	--Aligner.supp-aligns	<ul style="list-style-type: none"> • 0–4095

Name	Description	Command-Line Equivalent	Value
supp-as-sec	Determines if supplementary alignments should be reported with secondary flag.	--Aligner.supp-as-sec	<ul style="list-style-type: none"> • 0–1
supp-min-score-adj	Specifies amount to increase minimum alignment score for supplementary alignments. The score is computed by host software as 8 * match-score for DNA. The default is 0 for RNA.	--Aligner.supp-min-score-adj	
unclip-score	Specifies the score bonus for reaching the edge of the read.	--Aligner.unclip-score	<ul style="list-style-type: none"> • 0–127
unpaired-pen	Specifies the penalty for unpaired alignments, using Phred scale.	--Aligner.unpaired-pen	<ul style="list-style-type: none"> • 0–255

If you disable automatic detection of insert-length statistics via the `--enable-sampling` option, you must override all the following options to specify the statistics. For more information, see [Mean Insert Size Detection](#). The following options are part of the [Aligner] section of the configuration file.

Option	Description	Command-Line Equivalent	Value
pe-stat-mean-insert	Specifies the average template length.	--pe-stat-mean-insert	<ul style="list-style-type: none"> • 0–65535
pe-stat-mean-read-len	Specifies the average read length.	--pe-stat-mean-read-len	<ul style="list-style-type: none"> • 0–65535
pe-stat-quartiles-insert	Specifies a comma-delimited trio of numbers for the 25 th , 50 th , and 75 th percentile template lengths.	--pe-stat-quartiles-insert	<ul style="list-style-type: none"> • 0–65535

Option	Description	Command-Line Equivalent	Value
pe-stat-stddev-insert	Specifies the standard deviation of template length distribution.	--pe-stat-stddev- insert	• 0–65535

Variant Caller Options

The following options are in the Variant Caller section of the configuration file. For more information on the following options, see [Variant Caller Options on page 113](#).

Name	Description	Command-Line Equivalent	Value
dn-cnv-vcf	For <i>de novo</i> calling, filters joint structural variant VCF from the CNV calling step. If omitted, DRAGEN skips any checks with overlapping copy number variants.	--dn-cnv-vcf	
dn-input-vcf	For <i>de novo</i> calling, filters joint small variant VCF from the <i>de novo</i> calling step.	--dn-input-vcf	
dn-output-vcf	For <i>de novo</i> calling, specifies the file location for writing the filtered VCF file. If not specified, the input VCF is overwritten.	--dn-output-vcf	

Name	Description	Command-Line Equivalent	Value
dn-sv-vcf	For <i>de novo</i> calling, filters the joint structural variant VCF file from the SV calling step. If omitted, DRAGEN skips any checks with overlapping structural variants.	--dn-sv-vcf	
enable-joint-genotyping	To enable the joint genotyping caller, set to true.	--enable-joint-genotyping	<ul style="list-style-type: none"> • true • false
enable-multi-sample-gvcf	Enables generation of a multisample gVCF file. If set to true, DRAGEN requires a combined gVCF file as input.	--enable-multi-sample-gvcf	<ul style="list-style-type: none"> • true • false
enable-vlrd	Enables Virtual Long Read Detection.	--enable-vlrd	<ul style="list-style-type: none"> • true • false
pedigree-file	Specifies the path to a pedigree file that describes the familial relationships between panels (specific to joint calling). Only pedigree files that contain trios are supported.	--pedigree-file	
qc-snp-DeNovo-quality-threshold	Sets the threshold for counting and reporting <i>de novo</i> SNP variants.	--qc-snp-DeNovo-quality-threshold	
qc-indel-DeNovo-quality-threshold	Sets the threshold for counting and reporting <i>de novo</i> INDEL variants.	--qc-indel-DeNovo-quality-threshold	

Name	Description	Command-Line Equivalent	Value
variant	Specifies the path to a single gVCF file. You can use the --variant option multiple times to specify paths to multiple gVCF files. Use one file per line. Up to 500 gVCFs are supported.	--variant	
variant-list	Specifies the path to a file containing a list of input gVCF files that need to be combined. Use one file per line.	--variant-list	
vc-af-call-threshold	If the AF filter is enabled using --vc-enable-af-filter=true, the option sets the allele frequency call threshold for nuclear chromosomes to emit a call in the VCF. The default value is 0.01.	--vc-af-call-threshold	
vc-af-filter-threshold	If the AF filter is enabled using --vc-enable-af-filter=true, the option sets the allele frequency filter threshold for nuclear chromosomes to mark emitted VCF calls as filtered. The default value is 0.05.	--vc-af-filter-threshold	

Name	Description	Command-Line Equivalent	Value
vc-af-call-threshold-mito	If the AF filter is enabled using --vc-enable-af-filter-mito=true, the option sets the allele frequency call threshold to emit a call in the VCF for mitochondrial variant calling. The default value is 0.01.	--vc-af-call-threshold-mito	
vc-af-filter-threshold-mito	If the AF filter is enabled using --vc-enable-af-filter-mito=true, the option sets the allele frequency filter threshold to mark emitted VCF calls as filtered for mitochondrial variant calling. The default value is 0.02.	--vc-af-filter-threshold-mito	
vc-callability-normal-threshold	Specifies the normal sample coverage threshold for a site to be considered callable in the somatic callable regions report. The default value is 5.	--vc-callability-normal-thresh	
vc-callability-tumor-threshold	Specifies the tumor sample coverage threshold for a site to be considered callable in the somatic callable regions report. The default value is 50.	--vc-callability-tumor-thresh	

Name	Description	Command-Line Equivalent	Value
vc-clustered-sevent-penalty	SQ score penalty applied to phased clustered somatic events; set to 0 to disable the penalty. The default value is 4.0 for tumor-normal and 7.0 for tumor-only.	--vc-clustered-sevent-penalty	
vc-decoy-contigs	Specifies the path to a comma-separated list of contigs to skip during variant calling.	--vc-decoy-contigs	
vc-depth-annotation-threshold	Filters all non-PASS somatic (alt variants with a depth below this threshold. The default value is 0 (no filtering).	--vc-depth-annotation-threshold	
vc-depth-filter-threshold	Filters all somatic variants (alt or homref) with a depth below this threshold. The default value is 0 (no filtering).	--vc-depth-filter-threshold	
vc-emit-ref-confidence	Enables base pair resolution gVCF generation or banded gVCF generation.	--vc-emit-ref-confidence	<ul style="list-style-type: none"> • BP_RESOLUTION • GVCF
vc-enable-af-filter	Enables the allele frequency filter of nuclear chromosomes for somatic mode. The default value is false.	--vc-enable-af-filter	<ul style="list-style-type: none"> • true • false
vc-enable-af-filter-mito	Enables the allele frequency filter for mitochondrial variant calling. The default value is true.	--vc-enable-af-filter-mito	<ul style="list-style-type: none"> • true • false

Name	Description	Command-Line Equivalent	Value
vc-enable-baf	Enables B-allele frequency output. The default value is true.	--vc-enable-baf	<ul style="list-style-type: none"> • true • false
vc-enable-decoy-contigs	Enables variant calls on decoy contigs. The default value is false.	--vc-enable-decoy-contigs	<ul style="list-style-type: none"> • true • false
vc-enable-gatk-acceleration	Enables running variant caller in GATK mode. The default value is false.	--vc-enable-gatk-acceleration	<ul style="list-style-type: none"> • true • false
vc-enable-liquid-tumor-mode	Enables liquid tumor mode for tumor-normal analysis to account for tumor-in-normal contamination. The default value is false.	--vc-enable-liquid-tumor-mode	<ul style="list-style-type: none"> • true • false
vc-enable-non-homref-normal-filter	Enables the nonhomref normal filter, which filters out somatic variants if the normal sample genotype is not homozygous reference. The default value is true.	--vc-enable-non-homref-normal-filter	<ul style="list-style-type: none"> • true • false
vc-enable-orientation-bias-filter	Enables the orientation bias filter. The default value is false, which means the option is disabled.	--vc-enable-orientation-bias-filter	<ul style="list-style-type: none"> • true • false
vc-enable-phasing	Enables variants to be phased when possible. The default value is true.	--vc-enable-phasing	<ul style="list-style-type: none"> • true • false

Name	Description	Command-Line Equivalent	Value
vc-combine-phased-variants-distance	When the specified value is greater than 0, combines all phased variants in the phasing set that have a distance less than or equal to the provided value. The max allowed phasing distance is 15. The default value is 0, which disables the option.	--vc-combine-phased-variants-distance	0–15
vc-detect-systematic-noise	Enables a sensitive run mode for building a systematic noise file from normal samples. The mode is not intended for analyzing tumor samples. The default value is false.	--vc-detect-systematic-noise	<ul style="list-style-type: none"> • true • false
vc-enable-roh	Enables the ROH caller and output. The default value is true.	--vc-enable-roh	<ul style="list-style-type: none"> • true • false
vc-enable-triallelic-filter	Enables the multiallelic filter for somatic mode. The default value is false.	--vc-enable-triallelic-filter	<ul style="list-style-type: none"> • true • false

Name	Description	Command-Line Equivalent	Value
vc-enable-non-primary-allelic-filter	Similar to vc-enable-triallelic-filter, but less aggressive. Keep the allele per position with highest alt AD, and only filter the rest. The default is false. Not compatible with vc-enable-triallelic-filter.	--vc-enable-non-primary-allelic-filter	<ul style="list-style-type: none"> • true • false
vc-enable-vcf-output	Enables VCF file output during a gVCF run. The default value is false.	--vc-enable-vcf-output	<ul style="list-style-type: none"> • true • false
vc-enable-unequal-ntd-errors	Enables the Sample-specific SNV Error Estimation feature. The default value is true for somatic pipelines and false for germline pipelines.	--vc-enable-unequal-ntd-errors	<ul style="list-style-type: none"> • true • false
vc-enable-trimer-context	When enabled along with vc-enable-unequal-ntd-errors, DRAGEN uses trimer rather than monomer context to estimate SNV error rates. The default value is false, except when vc-enable-umi-liquid is enabled.	--vc-enable-trimer-context	<ul style="list-style-type: none"> • true • false

Name	Description	Command-Line Equivalent	Value
vc-forcegt-vcf	Forces genotyping for small variant calling. A file (*.vcf or *.vcf.gz) containing a list of small variants is required.	--vc-forcegt-vcf	*.vcf or *.vcf.gz file specifying the small variants to force genotype.
vc-gvcf-bands	Define bands for gVCF output. The default value is 1 10 20 30 40 60 80 for germline calling and 1 3 10 20 50 80 for somatic. If enable-multi-sample-gvcf is enabled, the default value is 5, 20, 60.	--vc-gvcf-bands	
vc-gvcf-homref-lod	Sets the limit of detection for somatic homref calls. The default value is 0.05.	--vc-gvcf-homref-lod	
vc-hard-filter	Uses a list of Boolean expressions to filter variant calls. The default expression is DRAGENHardQUAL:all: QUAL < 10.4139;LowDepth:all: DP < 1	--vc-hard-filter	<ul style="list-style-type: none"> • QD • MQ • FS • MQRankSum • ReadPosRankSum • QUAL • DP • GQ
vc-homref-depth-filter-threshold	In gvcf mode, filters all somatic homref variants with a depth below this threshold. The default value is 3.	--vc-homref-depth-filter-threshold	

Name	Description	Command-Line Equivalent	Value
vc-max-alternate-alleles	Specifies the maximum number of ALT alleles to output in a VCF or gVCF. The default value is 1000.	--vc-max-alternate-alleles	
vc-max-reads-per-active-region	Specifies the maximum number of reads for an active region for downsampling. The default value is 10000.	--vc-max-reads-per-active-region	
vc-max-reads-per-active-region-mito	Specifies the maximum number of reads for an active region of mitochondrial small variant calling. The default value is 40000.	--vc-max-reads-per-active-region-mito	
vc-max-reads-per-raw-region	Specifies the maximum number of reads per raw region for downsampling. The default value is 30000.	--vc-max-reads-per-raw-region	
vc-max-reads-per-raw-region-mito	Specifies the maximum number of reads covering a specified raw region of mitochondrial small variant calling. The default value is 40000.	--vc-max-reads-per-raw-region-mito	
vc-min-base-qual	Specifies the minimum base quality to be considered in the active region detection of the small variant caller. The default value is 10.	--vc-min-base-qual	

Name	Description	Command-Line Equivalent	Value
vc-min-contig-qual	Specifies the minimum base quality to be considered for De Bruijn graph construction. The default value is 10.	--vc-min-contig-qual	
vc-min-tail-qual	Specifies the minimum base quality to trim consecutive bases on either end of a read. The default value is 10.	--vc-min-tail-qual	
vc-min-call-qual	Specifies the minimum variant call quality for emitting a call. The default value is 3.	--vc-min-call-qual	
vc-min-read-qual	Specifies the minimum read quality (MAPQ) to be considered for small variant calling. The following default values exist: <ul style="list-style-type: none"> • 1 for germline • 3 for somatic T/N • 20 for somatic T-only 	--vc-min-read-qual	
vc-min-reads-per-start-pos	Specifies the minimum number of reads per start position for downsampling. The default value is 10.	--vc-min-reads-per-start-pos	
vc-min-tumor-read-qual	Specifies the minimum tumor read quality (MAPQ) to be considered for variant calling.	--vc-min-tumor-read-qual	

Name	Description	Command-Line Equivalent	Value
vc-override-tumor-pcr-params-with-normal	Ignores the tumor sample parameters and uses the normal sample parameters for analysis of both samples. The default value is true.	--vc-override-tumor-pcr-params-with-normal	<ul style="list-style-type: none"> • true • false
vc-remove-all-soft-clips	If set to true, the variant caller does not use soft clips of reads to determine variants. The default value is false.	--vc-remove-all-soft-clips	<ul style="list-style-type: none"> • true • false
vc-roh-blacklist-bed	If provided, the ROH caller ignores variants that are contained in any region in the block list BED.	--vc-roh-blacklist-bed	
vc-sq-call-threshold	Sets the SQ call threshold to emit a call in the VCF. The default value is 3.0 for tumor-normal and 0.1 for tumor-only.	--vc-sq-call-threshold	
vc-sq-filter-threshold	Sets the SQ filter threshold mark calls as filtered in the VCF. The default value is 17.5 for tumor-normal and 3.0 for tumor-only.	--vc-sq-filter-threshold	

Name	Description	Command-Line Equivalent	Value
vc-target-bed	If provided, the variant caller only outputs variants with a reference position that overlaps with any of the regions in the target BED.	--vc-target-bed	*.bed file
vc-target-bed-padding	Specifies a number of bases that the small variant caller then uses to pad each target BED region. The default value is 0.	--vc-target-bed-padding	
vc-target-coverage	Specifies the target coverage for downsampling. The default value is 500 for germline and 50 for somatic mode.	--vc-target-coverage	
vc-target-coverage-mito	Specifies the maximum number of reads with a start position overlapping any given position for mitochondrial small variant calling. The default value is 40000.	--vc-target-coverage-mito	

Name	Description	Command-Line Equivalent	Value
vc-target-vaf	<p>The vc-target-vaf is used to select the variant allele frequencies of interest. The variant caller will aim to detect variants with allele frequencies larger than this setting. We recommend adding a small safety factor, e.g. to ensure variants in the ballpark of 1% are detected, the minimum vc-target-vaf can be specified as 0.009 (0.9%). This setting will not apply a hard threshold, and it is possible to detect variants with allele frequencies lower than the selected threshold. On high coverage and clean datasets, a lower target-vaf may help increase sensitivity. On noisy samples (like FFPE) a higher target-vaf (like 0.03) maybe help reduce false positives. Using a low target-vaf may also increase runtime. Set the vc-target-vaf to 0 to disable this feature. When this feature is disabled the variant caller will require at least 2 supporting reads to discover a candidate variant.</p> <p>Default=0.01.</p>	--vc-target-vaf	[0, 1]

Name	Description	Command-Line Equivalent	Value
vc-tin-contam-tolerance	Sets the maximum tumor-in-normal contamination expected. Setting this to a nonzero value enables liquid tumor mode. If liquid tumor mode is enabled, the default value is 0.15. If liquid tumor mode is disabled, the default value is 0.	--vc-tin-contam-tolerance	
vc-excluded-regions-bed	Somatic mode only: If provided, variants that overlap with the regions in the BED file are hard-filtered and marked as "excluded_regions" in the filter column.	--vc-excluded-regions-bed	*.bed file
vc-systematic-noise	Specifies a BED file with site-specific systematic noise level to calculate AQ score (systematic noise score).	--vc-systematic-noise	
vc-systematic-noise-filter-threshold	Sets the AQ threshold for applying the systematic-noise filter. The default value is 10 for tumor-normal and 60 for tumor-only.	--vc-systematic-noise-filter-threshold	• 0–100

Name	Description	Command-Line Equivalent	Value
vc-enable-germline-tagging	Enable germline variant tagging using population databases. The default is false. Once enabled, it will also require user to specify Nirvana parameters. Details can be found in somatic small variant calling section.	--vc-enable-germline-tagging	<ul style="list-style-type: none"> • true • false
germline-tagging-db-threshold	The minimum alternative allele count in population database for a variant to be defined as germline. The default value is 50.	--germline-tagging-db-threshold	
germline-tagging-pop-af-threshold	The minimum population allele frequency for a variant to be defined as germline. Once specified, this will ignore the input from --germline-tagging-db-threshold.	--germline-tagging-pop-af-threshold	

Mutation Annotation Format (MAF) Conversion Options

Name	Description	Command-Line Equivalent	Value
enable-maf-output	Enables Mutation Annotation Format (MAF) output. The default value is false.	--enable-maf-output	<ul style="list-style-type: none"> • true • false

Name	Description	Command-Line Equivalent	Value
maf-transcript-source	Specifies desired transcript source for Mutation Annotation Format (MAF) output.	--maf-transcript-source	Refseq/Ensembl
maf-input-vcf	Specifies input VCF file for standalone Mutation Annotation Format (MAF) output.	--maf-input-vcf	*.hard-filtered.vcf.gz file
maf-input-json	Specifies input JSON file for standalone Mutation Annotation Format (MAF) output.	--maf-input-json	*.hard-filtered.vcf.annotated.json.gz file

CNV Caller Options

The following options are available for the CNV Caller.

Name	Description	Command-Line Equivalent	Value
cnv-exclude-bed	Specifies regions to exclude from CNV processing.	--cnv-exclude-bed	
cnv-exclude-bed-min-overlap	Specifies the minimum fraction of overlap between target intervals and the blocklist to exclude target from the list.	--cnv-exclude-bed-min-overlap	[0.0, 1.0]
cnv-cbs-alpha	Specifies the significance level for the test to accept change points. The default value is 0.01.	--cnv-cbs-alpha	

Name	Description	Command-Line Equivalent	Value
cnv-cbs-eta	Specifies the type I error rate of the sequential boundary for early stopping when using the permutation method. The default value is 0.05.	--cnv-cbs-eta	
cnv-cbs-kmax	Specifies the maximum width of smaller segment for permutation. The default value is 25.	--cnv-cbs-kmax	
cnv-cbs-min-width	Specifies the minimum number of markers for a changed segment. The default value is 2.	--cnv-cbs-min-width	[2,5]
cnv-cbs-nmin	Specifies the minimum length of data for maximum statistic approximation. The default value is 200.	--cnv-cbs-nmin	
cnv-cbs-nperm	Specifies the number of permutations used for p-value computation. The default value is 10000.	--cnv-cbs-nperm	
cnv-cbs-trim	Specifies the proportion of data to be trimmed for variance calculations. The default value is 0.025.	--cnv-cbs-trim	
cnv-counts-method	Specifies the overlap method for counting an alignment.	--cnv-counts-method	<ul style="list-style-type: none"> • midpoint • start • overlap
cnv-enable-gcbias-correction	Enables GC bias correction. The default is true.	--cnv-enable-gcbias-correction	<ul style="list-style-type: none"> • true • false
cnv-enable-gcbias-smoothing	Enables smoothing across GC bins. The default value is true.	--cnv-enable-gcbias-smoothing	<ul style="list-style-type: none"> • true • false
cnv-enable-gender-matched-pon	Enable gender matched PON normalization. The default value is true.	--cnv-enable-gender-matched-pon	<ul style="list-style-type: none"> • true • false

Name	Description	Command-Line Equivalent	Value
cnv-enable-ref-calls	When set to true, copy neutral (REF) calls are included in the output VCF.	--cnv-enable-ref-calls	<ul style="list-style-type: none"> • true • false
cnv-enable-self-normalization	Enables self-normalization.	--cnv-enable-self-normalization	<ul style="list-style-type: none"> • true • false
cnv-enable-tracks	Enables generation of track files that can be imported into IGV for viewing. The default is true.	--cnv-enable-tracks	<ul style="list-style-type: none"> • true • false
cnv-extreme-percentile	Specifies the extreme median percentile value used to filter out samples. The default value is 2.5.	--cnv-extreme-percentile	[0.0, 100.0]
cnv-filter-bin-support-ratio	If the span of supporting bins is less than the specified ratio with respect to the overall event length, the option filters out a candidate event. The default ratio is 0.2 (20% support).	--cnv-filter-bin-support-ratio	[0.0, 1.0]
cnv-filter-copy-ratio	Specifies the minimum copy ratio threshold value centered about 1.0 at which a reported event is marked as PASS in the output VCF file. The default value is 0.2.	--cnv-filter-copy-ratio	[0.0, 1.0]
cnv-filter-de-novo-quality	Sets the Phred-scale threshold for calling an event as <i>de novo</i> in the proband.	--cnv-filter-de-novo-quality	[0, inf]
cnv-filter-length	Specifies the minimum event length in bases at which a reported event is marked as PASS in the output VCF file. The default value is 10000.	--cnv-filter-length	[0, inf]

Name	Description	Command-Line Equivalent	Value
cnv-filter-limit-of-detection	Target limit of detection for enrichment somatic CNV alternative hypothesis test. The default value is 0.2.	--cnv-filter-limit-of-detection	[0.0, inf]
cnv-filter-qual	Specifies the QUAL value at which a reported event is marked as PASS in the output VCF file.	--cnv-filter-qual	[0, inf)
cnv-input	Specifies a CNV input file instead of a BAM. Files can be target.counts.gz or tn.tsv.gz for <i>de novo</i> .	--cnv-input	
cnv-interval-width	Specifies the width of the sampling interval for CNV WGS processing.	--cnv-interval-width	[100, inf)
cnv-max-percent-zero-samples	Specifies the number of zero coverage samples allowed for the target. If the target exceeds the specified threshold, then the target is filtered out. The default value is 5%.	--cnv-max-percent-zero-samples	[0.0, 100.0]
cnv-max-percent-zero-targets	Specifies the number of zero coverage targets allowed for the sample. If the sample exceeds the specified threshold, then the sample is filtered out. The default value is 5%.	--cnv-max-percent-zero-targets	[0.0, 100.0]
cnv-merge-distance	Specifies the maximum segment gap allowed for merging segments. The default value for Somatic WGS is 10k, for Germline WGS is 0. Default is inf when using CNV WES workflows.	--cnv-merge-distance	[0, inf)

Name	Description	Command-Line Equivalent	Value
cnv-merge-threshold	Specifies the maximum segment mean difference to merge two adjacent segments. The segment mean is represented as a linear copy ratio value.	--cnv-merge-threshold	[0.0, inf)
cnv-min-mapq	Specifies the minimum MAPQ for alignment to be counted.	--cnv-min-mapq	[1, inf)
cnv-normal-b-allele-vcf	Normal sample SNV VCF for determining het sites.	--cnv-normal-b-allele-vcf	
cnv-normal-cnv-vcf	Matched-normal CNV calls.	--cnv-normal-cnv-vcf	
cnv-normals-file	Specifies a single file to be used in the panel of normals. You can use the option multiple times, once for each file.	--cnv-normals-file	
cnv-normals-list	Specifies a text file containing paths to the list of reference target counts files to use as a panel of normals.	--cnv-normals-list	
cnv-num-gc-bins	Specifies the number of bins for GC bias correction. Each bin represents the GC content percentage. The default value is 25.	--cnv-num-gc-bins	<ul style="list-style-type: none"> • 10 • 20 • 25 • 50 • 100
cnv-num-singular-values	Number of singular values to retain for tangent normalization. The default is 5 when cnv-segmentation-mode=bed, otherwise dynamically detected.	--cnv-num-singular-values	[1, inf)

Name	Description	Command-Line Equivalent	Value
cnv-ploidy	Specifies the normal ploidy value. Used for estimating the copy number value emitted in the output VCF file. The default value is 2.	--cnv-ploidy	
cnv-population-b-allele-vcf	CNV population SNP input VCF file.	--cnv-population-b-allele-vcf	
cnv-qual-length-scale	Specifies the bias weighting factor to adjust QUAL estimates for segments with longer lengths. The default value is 0.9303 (2-0.1) and should not need to be modified.	--cnv-qual-length-scale	[0.0, 1.0]
cnv-qual-noise-scale	Specifies the bias weighting factor to adjust QUAL estimates based on sample variance. The default value is 1.0 and should not need to be modified.	--cnv-qual-noise-scale	[1.0, 10.0]
cnv-segmentation-mode	Specifies the segmentation algorithm to perform.	--cnv-segmentation-mode	<ul style="list-style-type: none"> • cbs • slm • hslm • aslm
cnv-skip-contig-list	A comma-separated list of contig identifiers to skip when generating intervals for WGS analysis. If not specified, the following contigs are skipped by default: chrM,MT,m,chrM.	--cnv-wgs-skip-contig-list	
cnv-slm-eta	Sets the baseline probability that the mean process changes its value. A higher value increases SLM segmentation sensitivity. The default value is 4e-5.	--cnv-slm-eta	[0, inf)

Name	Description	Command-Line Equivalent	Value
cnv-slm-fw	Specifies the minimum number of data points for a CNV to be emitted. The default value is 0.	--cnv-slm-fw	
cnv-slm-omega	Sets the scaling parameter modulating relative weight between experimental/biological variance. The default value is 0.3.	--cnv-slm-omega	[0, inf)
cnv-slm-stepeta	Specifies the distance normalization parameter. The default value is 10000. Only valid for HSMLM.	--cnv-slm-stepeta	[0, inf)
cnv-target-bed	Specifies a properly formatted BED file that indicates the target intervals to use for sample coverage. Use in WES analysis.	--cnv-target-bed	
cnv-target-factor-threshold	Specifies the bottom percentile of panel-of-normals medians to filter out useable targets. The default value is 1% for whole genome processing and 5% for targeted sequencing processing.	--cnv-target-factor-threshold	
cnv-truncate-threshold	Sets the percent threshold used to truncate extreme outliers. The default value is 0.1%.	--cnv-truncate-threshold	
cnv-use-somatic-vc-baf	Use somatic SNV BAFs from VC for B allele counting.	--cnv-use-somatic-vc-baf	
cnv-use-somatic-vc-vaf	Uses somatic SNV VAFs from VC to help determine purity and ploidy.	-cnv-use-somatic-vc-vaf	

Systematic Noise BED Creation Options

The following options are available to create systematic noise BED files from normal VCF files.

Name	Description	Command-Line Equivalent	Value
vc-systematic-noise-raw-input-list	Generates a list of VCFs to be used. One VCF per line.	--vc-systematic-noise-raw-input-list	
vc-systematic-noise-germline-vaf-threshold	Specifies the minimal variant allele frequency to remove potential germlines from the systematic noise file building. The default is unspecified, which indicates that all variants are used.	--vc-systematic-noise-germline-vaf-threshold	<ul style="list-style-type: none"> • 0–1
vc-systematic-noise-use-germline-tag	Determines whether to use DRAGEN internal germline tagging to remove potential germlines. Mutually exclusive with --vc-systematic-noise-germline-vaf-threshold. The default value is false. DRAGEN recommends setting this option to true for WGS analysis.	--vc-systematic-noise-use-germline-tag	<ul style="list-style-type: none"> • true • false
vc-systematic-noise-method	Describes the method used to calculate the systematic noise level across samples. The default method is mean.	--vc-systematic-noise-method	<ul style="list-style-type: none"> • mean • max • aggregate
vc-detect-systematic-noise	Enables a sensitive run mode for building a systematic noise file from normal samples. The mode is not intended for analyzing tumor samples. The default value is false.	--vc-detect-systematic-noise	<ul style="list-style-type: none"> • true • false

Structural Variant Caller Options

Name	Description	Command-Line Equivalent	Value
enable-sv	Enables the structural variant caller. The default value is false.	--enable-sv	<ul style="list-style-type: none"> • true • false
sv-call-regions-bed	Specifies a BED file containing the set of regions to call. Optionally, you can compress the file in GZIP or BZIP format.	--sv-call-regions-bed	
sv-denovo-scoring	Enables <i>de novo</i> quality scoring for structural variant joint diploid calling. Provide the pedigree file as well.	--sv-denovo-scoring	
sv-forcegt-vcf	Specifies a VCF of structural variants for forced genotyping. The variants are scored and included in the output VCF, even if not found in the sample data. The variants are merged with any additional variants discovered directly from the sample data.	--sv-forcegt-vcf	
sv-discovery	Enables SV discovery. Set to false when using <code>--sv-forcegt-vcf</code> to indicate that SV discovery should be disabled and only the forced genotyping input should be used.	--sv-discovery	<ul style="list-style-type: none"> • true • false

Name	Description	Command-Line Equivalent	Value
sv-exome	When set to true, configures the variant caller for targeted sequencing inputs, which includes disabling high depth filters. The default value is false.	--sv-exome	<ul style="list-style-type: none"> • true • false
sv-output-contigs	Set to true to have assembled contig sequences output in a VCF file. The default value is false.	--sv-output-contigs	<ul style="list-style-type: none"> • true • false
sv-region	Limits the analysis to a specified region of the genome for debugging purposes. You can use the option multiple times to build a list of regions.	--sv-region	<ul style="list-style-type: none"> • Must be in the format chr:startPos-endPos.

CYP2D6 Calling Options

Name	Description	Command-Line Equivalent	Value
enable-cyp2d6	Enables CYP2D6 diplotyping. The default value is false.	--enable-cyp2d6	<ul style="list-style-type: none"> • true • false

Repeat Expansion Detection Options

The following options can be set in the RepeatGenotyping section of the configuration file or on the command line. For more information, see [Repeat Expansion Detection with ExpansionHunter on page 228](#).

Name	Description	Command-Line Equivalent	Value
enable	Enables repeat expansion detection.	--repeat-genotype-enable	<ul style="list-style-type: none"> • true • false

Name	Description	Command-Line Equivalent	Value
specs	Specifies the full path to the JSON file that contains the repeat variant catalog (specification) describing the loci to call.	--repeat-genotype-specs	
use-catalog	Repeat variant catalog type to use (default - ~60 repeats, default_plus_smn - same as default with SMN repeat, expanded - ~50K repeats).	--repeat-genotype-use-catalog	<ul style="list-style-type: none"> • default • default_plus_smn • expanded

RNA-Seq Command Line Options

Name	Description	Command-Line Equivalent	Value
enable-rna	Enables processing of RNA-seq data.	--enable-rna	<ul style="list-style-type: none"> • true • false
annotation-file	Use to supply a gene annotation file. Required for quantification and gene-fusion.	--annotation-file, -a	<ul style="list-style-type: none"> • Path to GTF (or .gtf.gz) file
enable-rna-quantification	Enables RNA quantification.	--enable-rna-quantification	<ul style="list-style-type: none"> • true • false
rna-quantification-library-type	Specifies the type of the RNA-seq library. The default value is autodetect.	--rna-quantification-library-type	<ul style="list-style-type: none"> • IU • ISR • ISF • U • SR • SF • A
rna-quantification-gc-bias	Enables correction for GC bias in fragment counts.	--rna-quantification-gc-bias	<ul style="list-style-type: none"> • true • false
enable-rna-gene-fusion	Enables RNA gene fusion calling.	--enable-rna-gene-fusion	<ul style="list-style-type: none"> • true • false

Name	Description	Command-Line Equivalent	Value
rna-gf-restrict-genes	Ignores genes with biotype other than protein coding or lncRNA for gene fusions.	--rna-gf-restrict-genes	<ul style="list-style-type: none"> • true • false

UMI Options

Name	Description	Command Line Equivalent	Value
umi-library-type	Sets the batch option for correcting UMIs. Not required.	--umi-library-type	<ul style="list-style-type: none"> • random-duplex • random-simplex • nonrandom-duplex
umi-enable	Enables UMI-based read processing.	--umi-enable	<ul style="list-style-type: none"> • true • false
vc-enable-umi-solid	Enables solid tumor UMI-aware VC settings. The default value is false.	--vc-enable-umi-solid	<ul style="list-style-type: none"> • true • false
vc-enable-umi-liquid	Enables liquid tumor UMI-aware VC settings. The default value is false.	--vc-enable-umi-liquid	<ul style="list-style-type: none"> • true • false
vc-enable-umi-germline	Allow germline VC from UMI-collapsed reads. The default value is false.	--vc-enable-umi-germline	<ul style="list-style-type: none"> • true • false
umi-correction-scheme	Describes the methodology to use for correcting sequencing errors in UMIs.	--umi-correction-scheme	<ul style="list-style-type: none"> • lookup • random • none • positional
umi-correction-table	Provides the path to the correction table for lookup correction scheme.	--umi-correction-table	• Path to table file
umi-emit-multiplicity	Sets the consensus read output type.	--umi-emit-multiplicity	<ul style="list-style-type: none"> • both • duplex • simplex

Name	Description	Command Line Equivalent	Value
umi-min-supporting-reads	Specifies the number of input reads with matching UMI and position required to generate a consensus read.	--umi-min-supporting-reads	<ul style="list-style-type: none"> • Integer ≥ 1. The default is 2.
umi-metrics-interval-file	Provides the path to target regions file used for UMI on target metrics.	--umi-metrics-interval-file	<ul style="list-style-type: none"> • Path to valid BED file
umi-source	Specifies the location to read UMIs from.	--umi-source	<ul style="list-style-type: none"> • qname • bamtag • fastq
umi-fastq	Provides the path to a separate FASTQ file with UMI sequences for each read.	--umi-fastq	<ul style="list-style-type: none"> • Path to valid FASTQ file
umi-nonrandom-whitelist	Provides the path to a file containing valid nonrandom UMIs sequences. Enter one path per line.	--umi-nonrandom-whitelist	
umi-fuzzy-window-size	Collapses reads with matching UMIs and alignment positions up to the distance specified.	--umi-fuzzy-window-size	<ul style="list-style-type: none"> • Integer ≥ 1. The default is 3.

Systematic Noise BED Creation Options

The following options are applicable to create the systematic noise BED file from normal VCFs.

Step 1. Run DRAGEN somatic tumor-only pipeline on each of approximately 50 normal samples.

Name	Description	Command Line Equivalent	Value
vc-detect-systematic-noise	Runs the tumor-only pipeline in a very sensitive mode that aims to capture noise. Use --tumor-fastq1/2 or --tumor-bam-input to specify input reads. This step requires vc-enable-germline-tagging=true. To explicitly run without germline tagging, use vc-skip-germline-tagging=true. VCFs generated with this setting can be used in step 2 when building a systematic noise file from normal samples. This mode is not intended for analyzing tumor samples. The default value is false.	--vc-detect-systematic-noise	true false
vc-enable-germline-tagging	Enable germline variant tagging using population databases. The default is false. Once enabled, it will also require user to specify Nirvana parameters. Details can be found in somatic small variant calling section. When used with noise generation it helps prevent treating germline sites as noisy. It is strongly recommended to enable this option when generating VCFs for the normal samples.	--vc-enable-germline-tagging	true false

Step 2. Generate the final noise file with:

Name	Description	Command Line Equivalent	Value
build-sys-noise-vcfs-list	Path to text file containing list of normal VCF/GVCF files (from step 1) to be included in the systematic noise. One file per line.	--build-sys-noise-vcfs-list	
build-sys-noise-germline-vaf-threshold	Minimum variant allele frequency threshold to define germline variants. Variants with AF higher than this threshold will not contribute to the noise file. Set to 1 to disable.	--build-sys-noise-germline-vaf-threshold	Default=1. The valid range is [0-1].
build-sys-noise-use-germline-tag	germline-tag If available in the VCF use germline tags to prevent germline calls from contributing to systematic noise.	--build-sys-noise-use-germline-tag	Default=true
build-sys-noise-threads	Max number of threads used during noise generation. Each thread consumes approx. 70 GB of system memory.	--build-sys-noise-threads	Options are 1 or 2. Default=2.
build-sys-noise-method	Method to compute noise across samples ['mean'/'max']. For higher specificity 'max' is recommended, for higher sensitivity 'mean' is recommended.	--build-sys-noise-method	Default=mean

Name	Description	Command Line Equivalent	Value
build-sys-noise-decimal-precision	Number of decimal digits in noise file. Options are [3-6]. For typical WES/WGS with 50-500X coverage 3 decimal places should be sufficient. For deep UMI samples with low noise rates 5 decimal places are recommended. Lower precision may help reduce noise file size especially on WES/WGS. The default is set for accuracy.	--build-sys-noise-decimal-precision	Default=5
build-sys-noise-min-sample-cov	Min coverage at a site for a sample to be used towards noise estimation. At low coverages estimated allele frequencies become less reliable, but low coverage sites also tend to be noisy and useful for inclusion in the noise file.	--build-sys-noise-min-sample-cov	Default=5
build-sys-noise-min-supporting-samples	Min number of samples with noise at a position in order for a position to be considered systematic-noise	--build-sys-noise-min-supporting-samples	Default=2

CYP2D6 Options

Name	Description	Command Line Equivalent	Value
enable-cyp2d6	Enables CYP2D6 diplotyping. The default value is false.	--enable-cyp2d6	true false

Explify Options

There are two separate Explify capabilities available:

Explify analysis pipeline

Name	Description	Command Line Equivalent	Value
enable-explify	Enables the Explify Pipeline. The default value is false	--enable-explify	true false
explify-sample-list	Input sample list .tsv file with sample IDs, FASTQs, etc.	--explify-sample-list	See User Guide
explify-test-panel-name	Set to "RPIP""UPIP" for panel name	--explify-test-panel-name	RPIP UPIP
explify-test-panel-version	Set to test panel version (e.g. "7.3.2")	--explify-test-panel-version	See User Guide
explify-ref-db-dir	Path to root directory for Explify Database files	--explify-ref-db-dir	
explify-load-db-ram	Option to load database into RAM if not on ramdisk. The default value is false.	--explify-load-db-ram	true false
explify-no-read-qc	Option to turn off read QC on FASTQs before analysis. The default value is false.	--explify-no-read-qc	true false
explify-internal-control	Option to set internal control from an accepted list. The default value is "Enterobacteria phage T7"	--explify-internal-control	See User Guide
explify-internal-control-concentration	Option to set internal control concentration in copies/mL of sample. The default value is 12100000.	--explify-internal-control-concentration	Integer > 0
explify-ncpus	Option to set the number of CPUs available for processing	--explify-ncpus	[1,max avail]

Metagenomics kmer classifier

Name	Description	Command Line Equivalent	Value
enable-kmer-classifier	Enables the Kmer Classifier. The default value is false	--enable-kmer-classifier	true false

Name	Description	Command Line Equivalent	Value
kmer-classifier-input-read-file	Input sequence file (zipped or unzipped) to the Kmer Classifier	--kmer-classifier-input-read-file	
kmer-classifier-db-file	Database of sequences to classify against	--kmer-classifier-db-file	
kmer-classifier-load-db-ram	Load the database onto RAM. Do not use if database is on ramdisk. The default value is false	--kmer-classifier-load-db-ram	true false
kmer-classifier-multiple-inputs	Set to true to run with multiple inputs where input .tsv file has two columns: ID, Inputfile. The default value is false.	--kmer-classifier-multiple-inputs	true false
kmer-classifier-min-window	The minimum number of consecutive kmers for classify assignment at taxid. The default value is 1	--kmer-classifier-min-window	Integer >=1
kmer-classifier-output-read-seq	Option to enable read sequence column in the output file. The default value is false	--kmer-classifier-output-read-seq	true false
kmer-classifier-output-taxid-seq	Option to enable a taxid string column in the output file. The default value is false	--kmer-classifier-output-taxid-seq	true false
kmer-classifier-db-to-taxid-json	Path to JSON file that maps database IDs to external taxids, names, and ranks	--kmer-classifier-db-to-taxid-json	See User Guide
kmer-classifier-no-read-output	Option to not create individual read output. The default value is false	--kmer-classifier-no-read-output	true false
kmer-classifier-no-taxid-counts	Option to not write taxid count output file. The default value is false	--kmer-classifier-no-taxid-counts	true false

Name	Description	Command Line Equivalent	Value
kmer-classifier-protein-input	Option to indicate protein query sequences and database. The default value is false	--kmer-classifier-protein-input	true false
kmer-classifier-reads-to-keep	Path to file of read IDs to use in alignment with one read per line.	--kmer-classifier-reads-to-keep	See User Guide
kmer-classifier-no-remove-dups	Option to not deduplicate reads in input files	--kmer-classifier-no-remove-dups	true false
kmer-classifier-ncpus	Option to set the number of CPUs available for processing	--kmer-classifier-ncpus	[1,max avail]

Resources & References

The DRAGEN support pages on the [Illumina support site](#) provide additional resources. These resources include training, compatible products, and other considerations. Always check support pages for the latest versions.

Revision History

Document	Date	Description of Change
Document #200033181v02	April 2024	Added recipe for cfDNA with Enrichment on FFPE samples
Document #200033181v01	August 2023	<p>Updated BCL Data Conversion with the following:</p> <ul style="list-style-type: none"> • Added Reverse Complement information • Updated FASTQ Files information • Added Legacy Stats Output Files information <p>Updated known Limitations in the DRAGEN HLA Caller section</p>
Document #200033181v00	July 2023	<p>Added DRAGEN Recipes</p> <ul style="list-style-type: none"> • Germline WES • Germline WGS • Somatic UMI Tumor Normal • Somatic UMI Tumor Only • Somatic WES Tumor Normal • Somatic WES Tumor Only • Somatic WGS Tumor Normal • Somatic WGS Tumor Only <p>Added Explify pipeline</p> <ul style="list-style-type: none"> • Explify Analysis Pipeline • Kmer Classifier <p>Updated RNA QC Metrics info</p> <p>Updated the Ploidy Estimator section</p> <p>Updated CYP2B6 Output File section</p> <p>Added HBA Caller</p> <p>Added LPA Caller</p> <p>Added Rh Caller</p> <p>Added Targeted Variant Calling</p> <p>Added Population Haplotyping</p> <p>Updated Prepare a Custom Multigenome Reference</p> <p>Updated BCL Data Conversion</p> <p>ORA Compression and Decompression</p> <p>Updated Read Trimming Section</p>



Illumina, Inc.
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

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

© 2024 Illumina, Inc. All rights reserved.

illumina®