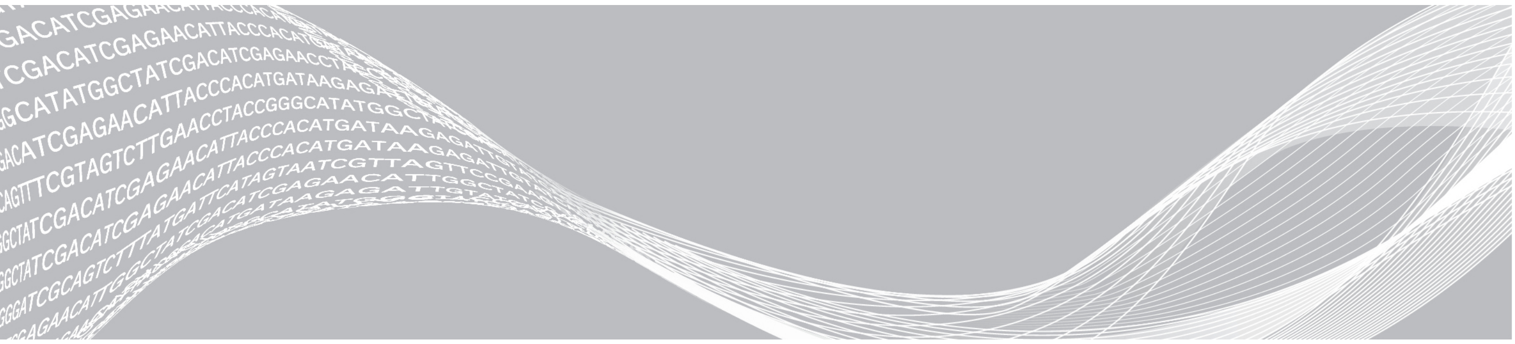
Illumina DRAGEN Bio-IT Platform v3.3

User Guide



Document # 1000000101034 v00

January 2020

Revision History

Document	Date	Description of Change
Document # 1000000101034 v00	January 2020	Added v3.3 to the document title. Changed the document number to 1000000101034.
Document # 1000000070494 v04	April 2019	Added version history for v03, added South Korean Technical Support phone number.
Document # 1000000070494 v03	April 2019	• Removed reference to --cram-reference option, which is no longer needed. • --annotation-sj-file option removed. Use --annotation-file instead. • Added information on the following options: • --vc-enable-phasing • --vc-tlod-filter-threshold • --vc-enable-triallelic-filter • --cnv-merge-distance • --cnv-enable-tracks • Added the following sections: • Down-sampling Options for Germline Small Variant Calling • Phasing and Phased Variants • SMA Calling • De Novo Quality Scoring • Multisample CNV Calling • Joint Segmentation • De Novo Calling Stage • Multisample CNV VCF Outuput • MapQ and BQ Coverage Filtering • Unique Molecular Identifiers (UMIs) • Gene Quantification • Updated the following sections: • Mitochondrial Calling • De Novo Joint Calling • Variant Hard Filtering • Repeat Genotyping • Structural Variant Calling • gVCF, Combine gVCF, and Joint VCF • Quality Scoring • Mapping and Aligning
Document # 1000000070494 v02	March 2019	Updated the following application names: • DRAGEN Genome Pipeline is now DRAGEN DNA Applications • DRAGEN Transcriptome Pipeline is now DRAGEN RNA Applications • DRAGEN Epigenome Pipeline is now DRAGEN Methylation Applications Repeat Genotyping is now Repeat Expansion Detection
Document # 1000000070494 v01	January 2019	Updated for version 3.2.5 of the software. Added Copy Number Variant Calling section. Added new options to Appendix A.
Document # 1000000070494 v00	December 2018	Initial release.

Table of Contents

Chapter 1 Illumina DRAGEN Bio-IT Platform

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:

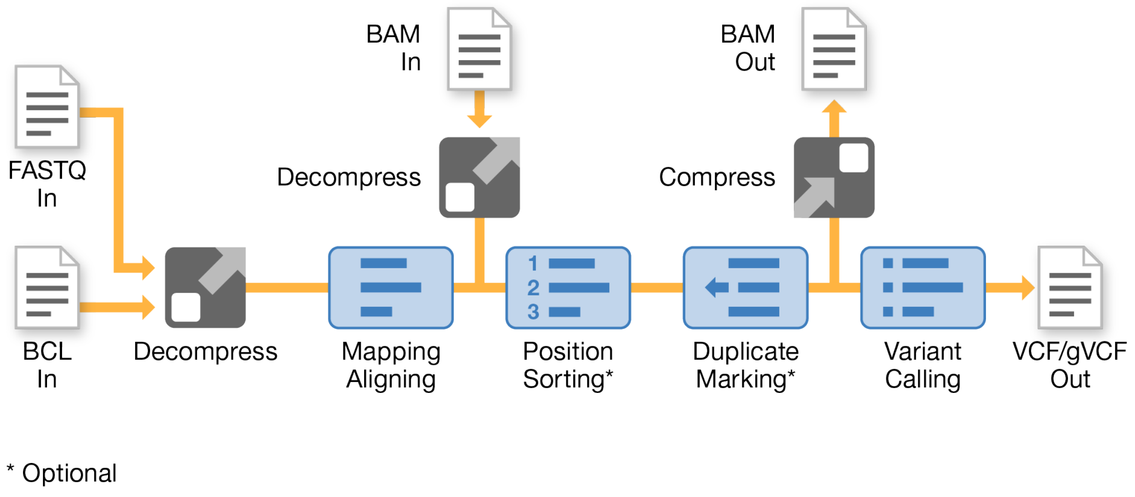
- Whole genome
- Exome
- RNA-Seq
- Methylome
- Cancer

All user 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.

DRAGEN DNA Applications

Figure 1 DRAGEN DNA Applications



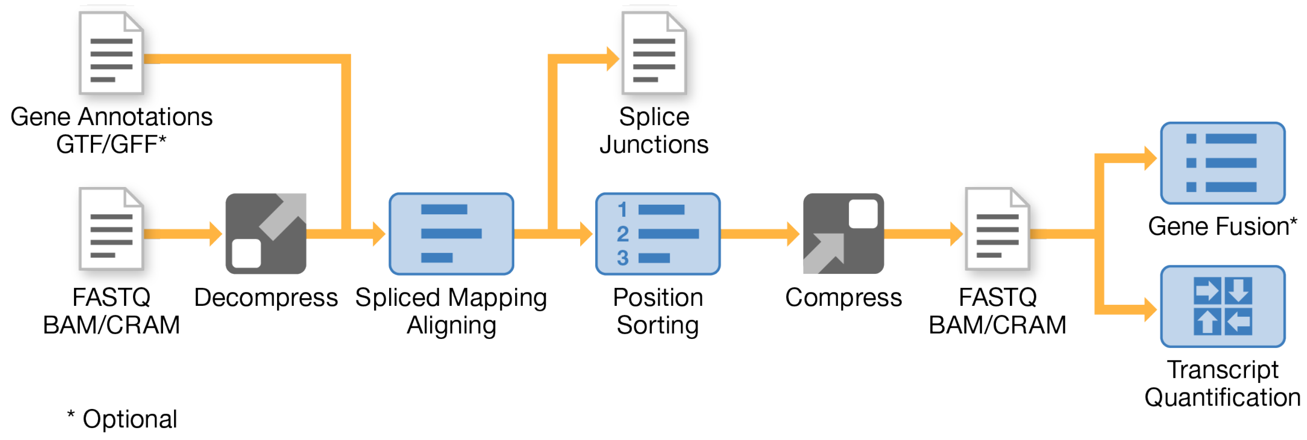
The DRAGEN DNA Applications massively accelerate the secondary analysis of NGS data. For example, the time taken to process an entire human genome at 30x coverage is reduced from approximately 10 hours (using the current industry standard, BWA-MEM+GATK-HC software) to approximately 20 minutes. Time scales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

These applications 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 compression and 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.

DRAGEN RNA Applications

The DRAGEN RNA Applications share many components with the DNA Applications, but the RNA-Seq read alignment stage is executed with a spliced aligner. Mapping of short seed sequences from RNA-Seq reads is performed similarly to the DRAGEN DNA Applications, but detecting splice junctions (the joining of noncontiguous exons in RNA transcripts) near the mapped seeds and incorporating the junctions into alignments of the entire read sequences are specific to the RNA Applications.



DRAGEN uses hardware accelerated algorithms to map and align RNA-Seq-based reads faster and more accurately than popular software tools. For instance, it can align 100 million paired-end RNA-Seq-based reads in about three minutes. With simulated benchmark RNA-Seq data sets, its splice junction sensitivity and specificity are unsurpassed.

The output BAM generated by the DRAGEN RNA Applications is compatible with popular downstream analysis tools.

DRAGEN Methylation Applications

The DRAGEN Methylation Applications provide support for automating the processing of bisulfite sequencing data, generating a BAM with the tags required for methylation analysis.

System Updates

DRAGEN is a flexible and extensible platform that is highly reconfigurable. Your DRAGEN subscription allows you to download updates to the DRAGEN processors and software. These updates provide speed, performance, throughput, and accuracy improvements.

Additional Resources and Support

For additional information, resources, system updates, and support, please visit the DRAGEN support page on the Illumina website.

Getting Started

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.

Running the System Check

After powering up, you can make sure that your DRAGEN system 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 system 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 powering up your DRAGEN system, 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

Loading the Reference Genome

Before a reference genome can be used with DRAGEN, it must be converted from FASTA format into a custom binary format for use with the DRAGEN hardware. For more information, see *Preparing a Reference Genome* on page 100.

Use the following command to load the hash table for your reference genome:

```
dragen -r <reference_hash-table_directory>
```

Make sure that the reference hash table directory is on the fast file IO drive.

The default location for the hash table for hg19 is as follows.

```
/staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149
```

The command to load reference genome hg19 from the default location is as follows.

```
dragen -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
```

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.

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.

Before running any analyses, the following are recommended:

- Copy your input data to /staging.
- Direct your outputs to /staging.
- Save a copy of the input data from the staging area in a separate location.

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 \
    -1 /staging/test/data/SRA056922.fastq \
    --output-directory /staging/test/output \
    --output-file-prefix SRA056922_dragen
```

For information on the command line options, see *DRAGEN Host Software* on page 6

Chapter 2 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 14. 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

The usage summary for the *dragen* command is as follows:

- **Build Reference/Hash Table**

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

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

```
dragen -r <REF_DIR> --output-directory <OUT_DIR> \
   --output-file-prefix <FILE_PREFIX> [options] -1 <FASTQ> \
   [-2 <FASTQ>] --enable-variant-caller true \
   --vc-sample-name <SAMPLE>
```

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

```
dragen -r <REF_DIR> --output-directory <OUT_DIR> \
   --output-file-prefix <FILE_PREFIX> [options] -1 <FASTQ> [-2 <FASTQ>]
```

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

```
dragen -r <REF_DIR> --output-directory <OUT_DIR> \
   --output-file-prefix <FILE_PREFIX> [options] -b <BAM> \
   --enable-variant-caller true --vc-sample-name <SAMPLE>
```

- **Run BCL Converter** *(BCL to *.fastq)*

```
dragen --bcl-conversion-only=true --bcl-input-dir <BCL dir> \
   --bcl-output-dir <output directory>
```

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

```
dragen -r <REF_DIR> --output-directory <OUT_DIR> \
   --output-file-prefix <FILE_PREFIX> [options] -1 <FASTQ> \
   [-2 <FASTQ>] --enable-rna true
```

For a complete list of command line options, see *Command Line Options* on page 117.

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 *Preparing a Reference Genome* on page 100. You must also specify the directory containing the preprocessed binary reference and hash tables with the *--ref-dir* option. This argument is always required.

You can load the reference genome and hash tables to DRAGEN card memory separately from processing reads, as follows.

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

You can use the *-l (--force-load-reference)* option to force the reference genome to load even if it is already loaded, as follows.

```
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.

Operating Modes

DRAGEN has two primary modes of operation, as follows:

- Mapper/aligner
- Variant caller

The DRAGEN system is capable of performing each mode independently or as an end-to-end solution. The DRAGEN system 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.

  Input is a mapped and aligned BAM file. DRAGEN produces a VCF file. If the BAM file is already sorted, sorting can be skipped by setting *--enable-sort* to *false*. 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. The DRAGEN Bio-IT Processor 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 and Cokus (aka directional and nondirectional) protocols, generating 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.

The remainder of this section provides more information on options that provide finer control of the DRAGEN pipeline.

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.

For brevity, 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. The user can control the output format from the map/align stage with the *--output-format <SAM|BAM|CRAM>* option. If the output file exists, the software 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 forces overwrite:

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

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

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

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

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

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

DRAGEN can generate mismatch difference (MD) tags, as described in the BAM standard. However, because there is a small performance cost to generating these strings, it is turned off by default. 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 only when a read has any suboptimal alignments qualifying for secondary output (even if none were output because *--Aligner.sec-aligns* was set to 0). Valid tag values are:

- 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. These tags provide alignment information (position, cigar, orientation) of groups of supplementary alignments, which are useful in structural variant calling.

Input Options

The DRAGEN system is capable of processing reads in FASTQ format or BAM/CRAM format. If your input FASTQ files end in .gz, DRAGEN automatically decompresses the files using hardware-accelerated decompression.

FASTQ Input Files

FASTQ input files can be single-ended or paired-end, as shown in the following examples.

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

  ```
  dragen -r <ref_dir> -1 <fastq> --output-directory <out_dir> \
     --output-file-prefix <out_prefix>
  ```

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

  ```
  dragen -r <ref_dir> -1 <fastq1> -2 <fastq2> \
     --output-directory <out_dir> --output-file-prefix <out_prefix>
  ```

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

  ```
  dragen -r <ref_dir> -1 <interleaved_fastq> -i
  ```

FASTQ samples can be segmented into multiple files to limit file size or to decrease the time to generate them. Both bcl2fastq and the DRAGEN BCL command use a common file naming convention, as follows:

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

For Example:

```
RDRS182520_S1_L001_R1_001.fastq.gz

RDRS182520_S1_L001_R1_002.fastq.gz

...

RDRS182520_S1_L001_R1_008.fastq.gz
```

The file naming convention is different for HiSeqX and NextSeq instruments.

These files do not need to be concatenated to be processed together by DRAGEN. To map/align any sample, provide it with 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 as well. This behavior can be turned off by setting *--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 may differ.

Input files must be located on a fast file system to avoid impact on system performance.

fastq-list input file

To provide multiple FASTQ input files, use the *--fastq-list <csv file name>* option to specify the name of a CSV file containing the list of FASTQ files. 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 the FASTQ input files to have arbitrary names, to be input from multiple subdirectories, and to have tags specified explicitly for multiple read groups. DRAGEN automatically generates a CSV file of the correct format during BCL conversion to FASTQ. This CSV file is named fastq_list.csv, and contains an entry for each FASTQ file or file pair (for paired-end data) produced during that run.

NOTE

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.

A CSV file of the correct format is automatically generated during BCL conversion to FASTQ using DRAGEN. This CSV file is named fastq_list.csv, and contains an entry for each fastq file or file pair (for paired-end data) produced during that run. ITo use this as a template for map-align input, copy the fastq_list.csv file and delete lines that do not correspond to inputs to the run being prepared.

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 values and should not contain white space or other extraneous characters.

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, leave empty if not paired-end)

Each FASTQ file referenced in the CSV list can be referenced only once. All values in the Read2File column must be either nonempty and reference valid files, or they 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:

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

Additional tags can be specified for each read group by adding a column title that is four characters, all-uppercase, and begins 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 two options 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 matching 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.

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
   AGAACGGA.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>
```

BAM input files

To use BAM files as input to the mapper/aligner, set *--enable-map-align* to true. If you leave this option set to false (the default), you can use the BAM file as input to the variant caller.

When you specify a BAM file as input, 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 would require you to re-sort the input data set by read name. DRAGEN vastly increases the speed of this operation by pairing the input reads, and sending them 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

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.

The *--cram-reference* option is no longer needed. The CRAM compressor and decompressor uses the DRAGEN reference.

The following options are used for providing 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
```

Handling of N bases

One of the techniques that DRAGEN uses for optimizing the handling of sequences can lead to the overwriting of 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, respectively, and they combine 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") that indicate which end of a pair the read represents. For BAM input with 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 has 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 87). 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 89) as an annotations file to help guide the aligner in a two-pass mode of operation.

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. The DRAGEN system 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, the DRAGEN software 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
```

Future versions of the DRAGEN system will allow FASTQ records to be individually tagged as belonging to different read groups.

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 [ACxxxxxxxxxxx] : used 1263.9 Gbases
   since 2018-Feb-15 (1263886160894 bases, unlimited)
   LICENSE_MSG|   Genome  bases [ACxxxxxxxxxxx] : 202000000
   LICENSE_MSG|   Genome  bases [total]         : 202000000
```

Autogenerated MD5SUM for BAMs

An MD5SUM file is generated automatically for BAM output files. This file has the same name as the BAM 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 BAM output file, which exactly matches the output of the Linux md5sum command.

The MD5SUM calculation is performed as the BAM 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).

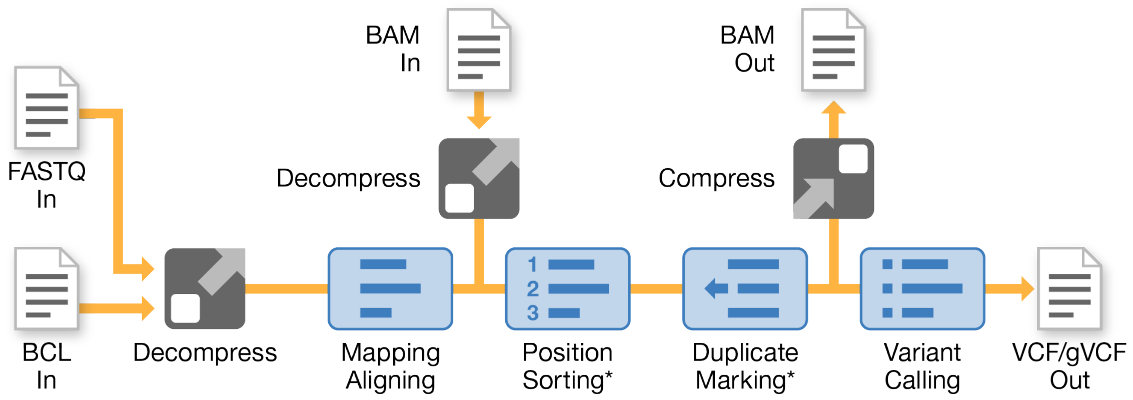
Configuration Files

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. You can override this file by using the *--config-file (-c)* option to specify a different file. The configuration file used for a given run supplies the default settings for that run, any of which can be overridden by command line options.

The recommended approach is to use the dragen-user-defaults.cfg file as a template to create default settings for different use cases. Copy dragen-user-defaults.cfg, rename the copy, then modify the new file for the specific use-case. Best practice is to put options that rarely change into the configuration file and to specify options that vary from run to run on the command line.

Chapter 3 DRAGEN DNA Applications

Figure 2 DNA Applications for DRAGEN



* Optional

DNA Mapping

Seed Density Option

The *seed-density* option controls how many (normally overlapping) primary seeds from each read the mapper looks up in its hash table for exact matches. The maximum density value of 1.0 generates a seed starting at every position in the read, ie, (L-K+1) K-base seeds from an L-base read.

Seed density must be between 0.0 and 1.0. Internally, an available seed pattern equal or close to the requested density is selected. The sparsest pattern is one seed per 32 positions, or density 0.03125.

- **Accuracy Considerations**—Generally, denser seed lookup patterns improve mapping accuracy. However, for modestly long reads (eg, 50 bp+) and low sequencer error rates, there is little to be gained beyond the default 50% seed lookup density.
- **Speed Considerations**—Denser seed lookup patterns generally slow down mapping, and sparser seed patterns speed it up. However, when the seed mapping stage can run faster than the aligning stage, a sparser seed pattern does not make the mapper much faster.

Relationship to Reference Seed Interval

Functionally, a denser or sparser seed lookup pattern has an impact very similar to a shorter or longer reference seed interval (build hash table option *--ht-ref-seed-interval*). 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 (the inverse of the reference seed interval) 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).

DRAGEN automatically adjusts its precise seed lookup pattern to ensure 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.

Map Orientations Option

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

The *--map-orientations* option can restrict the orientation of read mapping to only forward in the reference genome, or only reverse-complemented. The valid values for *--map-orientations* are as follows.

- 0–Either orientation (default)
- 1–Only forward mapping
- 2–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.

Seed-Editing Options

Although DRAGEN primarily maps reads by finding exact reference matches to short seeds, it 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. This is especially true when paired ends are used, because a seed match from either mate can successfully align the pair. But seed editing can, for example, 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 1 Seed Editing Options

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

edit-mode and edit-chain-limit

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

Mode	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 in edit modes 1 and 2 is a seed chain length test. Exact seeds are mapped to the reference in a first pass over a given read, and 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 is judged not to require seed editing, because there is already a promising 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* (including if 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-ended reads.

edit-seed-num and edit-read-len

For edit modes 1 and 2, when the heuristic triggers seed editing, these 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 much sparser 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, a greater increase in mapping accuracy can be obtained for the same time cost by editing seeds in sparse patterns for a large number of reads, than by editing seeds in dense patterns for a small number of 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. For example, with 21-base seeds, *edit-seed-num*=6 and *edit-read-len*=100, edited seeds can begin at offsets {0, 16, 32, 48, 64, 80} from the 5' end, consecutive seeds overlapping by 5 bases. Because sequencing technologies often yield better base qualities nearer the (5') beginning of each read, this can focus seed editing where it is most likely to succeed. When a particular read is shorter than *edit-read-len*, fewer seeds are edited.

Seed editing is more expensive when the reference seed interval (build hash table option *--ht-ref-seed-interval*) 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.

DNA Aligning

Smith-Waterman Alignment Scoring Settings

The first stage of mapping is to generate seeds from the read and look for exact matches in the reference genome. These results are then refined by running full Smith-Waterman alignments on the locations with the highest density of seed matches. This well-documented algorithm works by comparing each position of the read against all the candidate positions of the reference. These comparisons correspond to a matrix of potential alignments between read and reference. For each of these candidate alignment positions, Smith-Waterman generates scores that are used to evaluate whether the best alignment passing through that matrix cell reaches it by a nucleotide match or mismatch (diagonal movement), a deletion (horizontal movement), or an insertion (vertical movement). A match between read and reference provides a bonus, on the score, and a mismatch or indel imposes a penalty. The overall highest scoring path through the matrix is the alignment chosen.

The specific values chosen for scores in this algorithm indicate how to balance, for an alignment with multiple possible interpretations, 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. But any set of Smith-Waterman scoring parameters represents an imprecise model of genomic mutation and sequencing errors, and differently tuned alignment scoring values can be more appropriate for some applications.

The following alignment options control Smith-Waterman Alignment:

Command-Line Option Name	Configuration File Option Name
--Aligner.global	global
--Aligner.match-score	match-score
--Aligner.match-n-score	match-n-score
--Aligner.mismatch-pen	mismatch-pen
--Aligner.gap-open-pen	gap-open-pen
--Aligner.gap.ext.pen	gap-ext-pen
--Aligner.unclip-score	unclip-score
--Aligner.no-unclip-score	no-unclip-score
--Aligner.aln-min-score	aln-min-score

- *global*

  The *global* option (value can be 0 or 1) controls whether alignment is forced to be end-to-end in the read. When set to 1, alignments are always end-to-end, as in the Needleman-Wunsch global alignment algorithm (although not end-to-end in the reference), and 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 nonnegative.

  Generally, *global*=0 is preferred for longer reads, so significant read segments after a break of some kind (large indel, structural variant, chimeric read, and so forth) can be clipped without severely decreasing the alignment score. Setting *global=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, with *global*=0, multiple (chimeric) alignments can be reported when various portions of a read match widely separated reference positions.

  Using *global*=1 is sometimes preferable with short reads, which 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.

  Consider using the *unclip-score* option, or increasing it, instead of setting *global*=1, to make a soft preference for unclipped alignments.

- *match-score*

  The *match-score* option is the score for a read nucleotide matching a reference nucleotide (A, C, G, or T). Its value is an unsigned integer, from 0 to 15. *match_score*=0 can only be used when *global*=1. A higher match score results in longer alignments, and fewer long insertions.

- *match-2-score*

  The match-2-score option is the score for a read nucleotide matching a 2-base IUPAC-IUB code in the reference (K, M, R, S, W, or Y). This option is a signed integer, ranging from -16 to 15.

- *match-3-score*

  The *match-3-score* option is the score for a read nucleotide matching a 3-base IUPAC-IUB code in the reference (B, D, H, or V). This option is a signed integer, from -16 to 15.

- *match-n-score*

  The *match-n-score* option is the score for a read nucleotide matching an N code in the reference. This option is a signed integer, from -16 to 15.

- *mismatch-pen*

  The *mismatch-pen* option is the penalty (negative score) for a read nucleotide mismatching any reference nucleotide or IUPAC-IUB code (except ‘N’, which cannot mismatch). This 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.

- *gap-open-pen*

  The *gap-open-pen* option is the penalty (negative score) for opening a gap (ie, an insertion or deletion). This value is only for a 0-base gap. It is always added to the gap length times gap-ext-pen. This 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, with clipping or alignment through SNPs used instead.

- *gap-ext-pen*

  The *gap-ext-pen* option is the penalty (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, with short indels, clipping, or alignment through SNPs used instead.

- *unclip-score*

  The *unclip-score* option is the score bonus for an alignment reaching the beginning or end of the read. An end-to-end alignment receives twice this bonus. This option is an unsigned integer, from 0 to 127. A higher unclipped bonus causes alignment to reach the beginning and/or end of a read more often, where this can be done without too many SNPs or indels.

  A nonzero unclip-score is useful when *global=0* to make a soft preference for unclipped alignments. Unclipped bonuses have little effect on alignments when *global=1*, because end-to-end alignments are forced anyway (although 2 × unclip-score does add to every alignment score unless *no-unclip-score = 1*). It is recommended to use the default for unclip-score when *global=1*, because some internal heuristics consider how local alignments would have been clipped.

  Note that, especially with longer reads, setting unclip-score much higher than gap-open-pen can have the undesirable effect of insertions at or near one end of a read being utilized as pseudoclipping, as happens with global=1.

- *no-unclip-score*

  The *no-unclip-score* option can be 0 or 1. The default is 1. When *no-unclip-score* is set to 1, any unclipped bonus (unclip-score) contributing to an alignment is removed from the alignment score before further processing, such as comparison with *aln-min-score*, comparison with other alignment scores, and reporting in AS or XS tags. However, the unclipped bonus still affects the best-scoring alignment found by Smith-Waterman alignment to a given reference segment, biasing toward unclipped alignments.

When unclip-score > 0 causes a Smith-Waterman local alignment to extend out to one or both ends of the read, the alignment score stays the same or increases if *no-unclip-score*=0, whereas it stays the same or decreases if *no-unclip-score*=1.

The default, *no-unclip-score*=1, is recommended when *global*=1, because every alignment is end-to-end, and there is no need to add the same bonus to every alignment.

When changing *no-unclip-score*, consider whether *aln-min-score* should be adjusted. When *no-unclip-score*=0, unclipped bonuses are included in alignment scores compared to the *aln-min-score* floor, so the subset of alignments filtered out by *aln-min-score* can change significantly with *no-unclip-score*.

- *aln-min-score*

  The *aln-min-score* option specifies a minimum acceptable alignment score. Any alignment results scoring lower are discarded. Increasing or decreasing *aln-min-score* can reduce or increase the percentage of reads mapped. This option is a signed integer (negative alignment scores are possible with *global*=0).

  aln-min-score also affects MAPQ estimates. The primary contributor to MAPQ calculation is the difference between the best and second-best alignment scores, and *aln-min-score* serves as the suboptimal alignment score if nothing higher was found except the best score. Therefore, increasing *aln-min-score* can decrease reported MAPQ for some low-scoring alignments.

Paired-End Options

DRAGEN can process paired-end data, passed in either 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:

- *Reorientation*

  The *pe-orientation* option specifies the expected paired-end orientation. Only pairs with this orientation can be flagged as proper pairs. Valid values are as follows:

  - 0–FR (default)
  - 1–RF
  - 2–FF

- *unpaired-pen*

  For paired end reads, best mapping positions are determined jointly for each pair, 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. This 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, it is scaled into alignment score space based on Smith-Waterman scoring parameters.

- *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 mapping single-ended, plus this value. 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 and *pe-max-penalty* affects only reported MAPQ for paired alignments.

Mean Insert Size Detection

When working with paired-end data, DRAGEN must choose among the highest-quality alignments for the two ends to try to choose likely pairs. To make this choice, DRAGEN uses a Gaussian statistical model to evaluate the likelihood that a pair of alignments constitutes a pair. This model is based on the intuition that a particular library prep tends to create fragments of roughly similar size, thus producing pairs whose insert lengths cluster well around some mean insert length.

If you know the statistics of your library prep for an input file (and the file consists of a single read group), you can specify the characteristics of the insert-length distribution: mean, standard deviation, and three quartiles. These characteristics can be specified with the *Aligner.pe-stat-mean-insert*, *Alinger.pe-stat-stddev-insert*, *Aligner.pe-stat-quartiles-insert*, and *Aligner.pe-stat-mean-read-len* options. However, it is typically preferable to allow DRAGEN to detect these characteristics automatically.

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 in a report, as follows:

```
Final paired-end statistics detected for read group 0, based on 79935
   high quality pairs for FR orientation
   Quartiles (25 50 75) = 398 410 421
   Mean = 410.151
   Standard deviation = 14.6773
   Boundaries for mean and standard deviation: low = 352, high = 467
   Boundaries for proper pairs: low = 329, high = 490
   NOTE: DRAGEN's insert estimates include corrections for clipping (so
      they are no identical to TLEN)
```

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 100000 reads are unmapped or if all pairs are improper pairs, then the standard deviation is set to 10000 and the mean and quartiles are set to 0. Note that the minimum value for standard deviation is 12, which is independent of the number of samples.

For RNA-Seq 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. This estimate leads to a more accurate mean and standard deviation for RNA-Seq data and proper pairing.

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. Normally, the DRAGEN host software sets the rescue radius to 2.5 standard deviations of the empirical insert distribution. But in cases where the insert standard deviation is large compared to the read length, the rescue radius is restricted to limit mapping slowdowns. In this case, 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 up to four independent alignments for each read. These alignments can be chimeric, meaning they map different regions of the read, or they may be suboptimal mappings of the read to different areas of the alignment.

As DRAGEN tracks the four best alignments for a read, it gives priority to chimeric (supplemental) alignments over suboptimal (secondary). In other words, it tracks as many chimeric alignments as possible up to its limit of four. If there are fewer than four chimeric alignments, then the remaining slots are used to track suboptimal alignments

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

- *mapq-max*

  The *mapq-max* option specifies a ceiling on the estimated MAPQ that can be reported for any alignment, from 0 to 255. If the calculated MAPQ is higher, this value is reported instead. The default is 60.

- *supp-aligns*, *sec-aligns*

  The *supp-aligns* and *sec-aligns* 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 31 alignments are reported for any read total, including primary, supplementary, and secondary. Therefore, *supp-aligns* and *sec-aligns* each range from 0 to 30. Supplementary alignments are tracked and output with higher priority than secondary ones.

High settings for these two options impact speed so it is advisable to increase only as needed.

- *sec-phred-delta*

  The *sec-phred-delta* option controls which secondary alignments are emitted based on the alignment score relative to the primary reported alignment. Only secondary alignments with likelihood within this Phred value of the primary are reported.

- *sec-aligns-hard*

  The *sec-aligns-hard* option suppresses the output of all secondary alignments if there are more secondary alignments than can be emitted. Set *sec-aligns-hard* to 1 to force the read to be unmapped when not all secondary alignments can be output.

- *supp-as-sec*

  When the *supp-as-sec* option is set to 1, then supplementary (chimeric) alignments are reported with SAM FLAG 0x100 instead of 0x800. The default is 0. The *supp-as-sec* option provides compatibility with tools that do not support FLAG 0x800.

- *hard-clips*

  The *hard-clips* option is used as a field of 3 bits, with values ranging from 0 to 7. The bits specify alignments, as follows:

  - Bit 0–primary alignments
  - Bit 1–supplementary alignments
  - Bit 2–secondary alignments

  Each bit determines whether local alignments of that type are reported with hard clipping (1) or soft clipping (0). The default is 6, meaning primary alignments use soft clipping and supplementary and secondary alignments use hard clipping.

ALT-Aware Mapping

The GRCh38 human reference contains many more alternate haplotypes (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. DRAGEN ALT-aware mapping eliminates this issue, and instead obtains both sensitivity and specificity improvements from ALT contigs.

ALT-aware mapping requires hash tables that are built with ALT liftover alignments specified (see *ALT-Aware Hash Tables* on page 102). If a hash table built with liftover alignments is provided, DRAGEN automatically runs with ALT-aware mapping. Set the *--alt-aware* option to false to disable ALT-Aware mapping with a liftover reference.

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

When ALT-aware mapping is enabled, 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.

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

- Mapping without ALT contigs in the reference:
  - False-positive variant calls result when reads matching an alternate haplotype misalign somewhere else.
  - 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:
  - False-positive variant calls from misaligned reads matching ALT contigs are eliminated.
  - Low or zero aligned coverage in primary assembly regions covered by alternate haplotypes, due to some reads 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 dramatically reduced throughout regions covered by alternate haplotypes.
- Mapping with ALT contigs and alt awareness:
  - False-positive variant calls from misaligned reads matching ALT contigs are eliminated.
  - Normal aligned coverage in regions covered by alternate haplotypes because primary alignments are 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. The *--enable-sort* option can be used to enable or disable creation of the BAM file, as follows:

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

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

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 produces a BAM file with duplicates marked in the FLAG field, or with duplicates entirely removed.

In testing, enabling duplicate marking adds minimal run time over and above the time required to produce the sorted BAM file. The additional time is approximately 1-2 minutes for a 30x whole human genome, which is a huge improvement over the long run times of open source tools.

The Duplicate Marking Algorithm

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

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

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

In addition, an unpaired read may be marked as a duplicate if it has identical coordinate and orientation with either end of any other read, whether paired or not.

Unmapped or secondary alignments are never marked as duplicates.

When DRAGEN has identified a group of duplicates, it picks one as the best of the group, and marks the others with the BAM PCR or optical duplicate flag (0x400, or decimal 1024). For this 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 idea of this score is to try, all other things being equal, to preserve the reads with the highest-quality base calls.

If two reads (or pairs) 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 this attribute, then DRAGEN chooses a winner arbitrarily.

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

$$score\,(R) = \frac{\sum_i (R.QUAL[i]\ where\ R.QUAL[i] \geq dedup\_min\_qual)}{sequence\_length(R)}$$

Where R is a BAM record, QUAL is its array of Phred quality scores, and *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.

This score is stored as a one-byte number, with values rounded down to the nearest one-quarter. This rounding may lead to different duplicate marks from those chosen by Picard, but because the reads were very close in quality this has negligible impact on variant calling results.

Duplicate Marking Limitations

The limitations to DRAGEN duplicate marking implementation are as follows:

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

Duplicate Marking Settings

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

- *--enable-duplicate-marking*

  Set to true to enable duplicate marking. When *--enable-duplicate-marking* is enabled, the output is sorted, regardless of the value of the *enable-sort* option.
- *--remove-duplicates*

  Set to true to suppress the output of duplicate records. If set to false, set the 0x400 flag in the FLAG field of duplicate BAM records. When *--remove-duplicates* is enabled, then *enable-duplicate-marking* is forced to enabled as well.
- *--dedup-min-qual*

  Specifies the Phred quality score below which a base should be excluded from 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 approach taken includes performing localized de novo assembly in regions of interest to generate candidate haplotypes and performing read likelihood calculations using a hidden Markov model (HMM).

Variant calling is disabled by default. You can enable variant calling by setting the *--enable-variant-caller* option to true.

The Variant Caller Algorithm

The DRAGEN Haplotype Caller performs the following steps:

- **Active Region Identification**—Areas where multiple reads disagree with the reference are identified, and windows around them (active regions) are selected for processing.
- **Localized Haplotype Assembly**—In each active region, all the overlapping reads are assembled into a de Bruijn graph (DBG), 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 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.
- **Haplotype Alignment**—Each extracted haplotype is Smith-Waterman aligned back to the reference genome, to determine what variations from the reference it implies.
- **Read Likelihood Calculation**—Each read is tested 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**—The possible diploid combinations of variant events from the candidate haplotypes are formed, and for each of them, a conditional probability of observing the entire read pileup is calculated, using the constituent probabilities of observing each read given each haplotype from the pair HMM evaluation. These feed into the Bayesian formula to calculate a likelihood that each genotype is the truth, given the entire read pileup observed. Those genotypes with maximum likelihood are reported.

Variant Caller Options

The following options control the variant caller stage of the DRAGEN host software.

- *--enable-variant-caller*

  Set *--enable-variant-caller* to *true* to enable the variant caller stage for the DRAGEN pipeline.
- *--vc-target-bed*

  Restricts processing to regions specified in the BED file.
- *--vc-sample-name*

  The *--vc-sample-name* option specifies the sample name being processed. This field is required when running the variant caller in stand-alone mode. In end-to-end mode, you can use this option or use the *--RGSM map/align* option and that option is passed through to the variant caller.
- *--vc-target-coverage*

  The *--vc-target-coverage* option specifies the target coverage for downsampling. The default value is 500 for germline mode and 1000 for somatic mode.
- *--vc-enable-gatk-acceleration*

  If *--vc-enable-gatk-acceleration* is set to true, the variant caller runs in GATK mode (concordant with GATK 4.0).
- *--vc-remove-all-soft-clips*

  If *--vc-remove-all-soft-clips* is set to true, the variant caller does not use soft clips of reads to determine variants.
- *--vc-decoy-contigs*

  The *--vc-decoy-contigs* option 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*

  If *--vc-enable-decoy-contigs* is set to true, variant calls on the decoy contigs are enabled. The default value is false.
- *--vc-enable-phasing*

  The *-vc-enable-phasing* option enables variants to be phased when possible. The default value is true.

Down-sampling Options for Germline Small Variant Calling

The options for down-sampling reads in the germline small variant calling pipeline are as follows:

- *--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.

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 example that shows that the DP reported in a variant record can far exceed the *--vc-target-coverage* default value (of say 500):

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, its reported depth could be as high as 1200. This example shows that the DP reported in a variant record can far 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 down-sampling 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 down-sampling method.

The down-sampling 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 down-sampling 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 ensure that there is always at least a minimum number of reads (set to *--vc-min-reads-per-start-pos*) at any start position.

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

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, that identifies which set the phased variant is in. The value in the field in 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;FractionInformativeReads=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+0
1,8.387e-
05,5.300e+01,4.500e+02,4.500e+02,4.500e+02:0.00,34.77,37.77,34.77,69.54,37.77:11
,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;FractionInformativeReads=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+0
1,7.067e-
05,5.204e+01,4.500e+02,4.500e+02,4.500e+02:0.00,34.77,37.77,34.77,69.54,37.77:10
,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.

Mitochondrial Calling

For the small variant calling processing of the mitochondrial chromosome, significant changes were made starting in DRAGEN version 3.2 compared to previous versions.

In previous versions, chrM was either handled as diploid (if no sex was provided on the command line) or haploid (if sex was provided on the command line). Due to the nature of chromosome M, neither a haploid or a diploid model is adequate because a given cell has many copies of the haploid mitochondrial chromosome and these mitochondria copies don't share the exact same DNA sequence. Typically, there are approximately 100 mitochondria in each mammalian cell, and 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%.

Starting in DRAGEN 3.2, chrM is now processed through a continuous AF pipeline, which is similar to the somatic variant calling pipeline. In this case, a single ALT allele is considered, and the AF is estimated, and can be anywhere between 0% and 100%.

The mitochondrial chromosome processing is now more accurate than in previous versions, because in DRAGEN versions prior to 3.2, low AF calls were not output (due to expecting the AF being close to 100% in a haploid model). In the current version, you will be able to see all the single ALT allele variants on the mitochondrial chromosome, across the whole range of AF (from low AF to high AF).

QUAL is not output in the chrM variant records. Instead, the confidence score is INFO/LOD.

```
##INFO=<ID=LOD,Number=1,Type=Float,Description="Variant LOD score">
```

which gives the confidence that a variant is present at a given locus.

GQ is not output in the chrM variant records, because we don't test for multiple diploid genotype candidates. Instead, an ALT allele is considered as a candidate variant, and if FORMAT/LOD>emit_threshold (default = 4), then the FOMAT/GT is hard-coded to "0/1", and the FORMAT/AF yields an estimate on the variant allele frequency, which range is anywhere within [0,1].

- If FORMAT/LOD > emit_threshold, the variant is not emitted in the VCF
- If FORMAT/LOD > emit_threshold but FORMAT/LOD < lod_fstar_threshold the variant is emitted in the VCF but FILTER=lod_fstar.
- If FORMAT/LOD > emit_threshold and FORMAT/LOD > lod_fstar_threshold the variant is emitted in the VCF and FILTER=PASS.

The following are example VCF records on the chrM, showing one call with very high AF and another with very low AF. In both cases FORMAT/LOD > emit_threshold. FORMAT/LOD is also > lod_fstar threshold, so the FILTER annotation is PASS.

```
chrM 2259 . C T . PASS
   DP=9791;MQ=60.00;LOD=38838.40;FractionInformativeReads=0.994
   GT:AD:AF:F1R2:F2R1:DP:SB:MB
   0/1:5,9729:0.999:1,5007:4,4722:9734:3,2,4885,4844:1,4,4807,4922
```

```
chrM 16192 . C T . PASS
   DP=9644;MQ=60.00;LOD=26.12;FractionInformativeReads=0.992
   GT:AD:AF:F1R2:F2R1:DP:SB:MB
   0/1:9537,26:0.003:5530,16:4007,10:9563:4484,5053,13,13:4961,4576,19,
```

gVCF and joint VCF mode

In gVCF mode, the NON_REF regions are output alongside the variant records. The following is an example of NON_REF and variant regions output in chrM gVCF mode:

```
chrM 751 . A <NON_REF> . PASS END=1437 GT:AD:DP:GQ:MIN_DP:PL:SPL:ICNT
   0/0:6920,9:6929:99:4077:0,120,1800:0,255,255:40,4
```

```
chrM 1438 . A G,<NON_REF> . PASS
   DP=8500;MQ=57.39;LOD=30441.87;FractionInformativeReads=0.871
   GT:AD:AF:F1R2:F2R1:DP:SB:MB
   0/1:0,7400,0:1.000:0,3765,0:0,3635,0:7400:0,0,3994,3406:0,0,3633,3767
```

```
chrM 1439 . A <NON_REF> . PASS END=2258 GT:AD:DP:GQ:MIN_DP:PL:SPL:ICNT
   0/0:6120,10:6130:99:4190:0,120,1800:0,255,255:40,14
```

FORMAT/GT

In NON_REF regions, the FORMAT/GT is hard-coded to 0/0 and at variant loci, the FORMAT/GT is hard-coded to 0/1.

The FORMAT/GT for chrM is not determined by FORMAT/AF. It is determined by whether a variant is emitted at a position or not. The decision to emit the variant or not is determined by comparing the INFO/LOD score to a threshold. All variants with FORMAT/LOD > emit_threshold get emitted. Variants with FORMAT/LOD < emit_threshold are not emitted in the gVCF and get banded in a NON_REF region with FORMAT/GT=0/0.

One of the following two scenarios can occur at a given position.

- A variant is detected by the genotyper at a given position, and the FORMAT/LOD > emit_threshold. In this case, the FORMAT/GT is hard-coded to 0/1, and DRAGEN outputs FORMAT/AD, FORMAT/DP and FORMAT/AF computed at that position.
- No variant is detected at the given position, or, a variant is detected but the FORMAT/LOD < emit_threshold. In this case, the FORMAT/GT is hard-coded to 0/0, and the position gets banded with contiguous positions if they are in the same scenario. All positions with FORMAT/GT = 0/0 get banded together. The FORMAT/DP reported for the band is computed as the median DP across all

positions within the band. The FORMAT/AD values reported in the band are the AD values selected at the position where FORMAT/DP = median DP. In the joint VCF, FORMAT/AF is computed based on FORMAT/AD.

The following are examples of variant record on chrM in a trio joint VCF.

The first sample shows a variant, but FORMAT/FT is lod_fstar because FORMAT/LOD < lod_fstar_ threshold.

The second sample does not have a variant and the FORMAT/AF=0.

The third sample does not have a variant even though the FORMAT/AF>0, this means that that position for the sample is in a NON_REF region over which no variant was detected with sufficient confidence.

```
chrM 2622 . G A . . DP=11199;MQ=59.73 GT:AD:AF:DP:FT:LOD:F1R2:F2R1
   0/1:3375,8:0.002:3383:lod_fstar:4.4:1689,2:1686,6
   0/0:5629,1:0:5118:PASS:.:.:. 0/0:3505,8:0.003:2645:PASS:.:.:.
```

Somatic mode

The DRAGEN Somatic Pipeline allows ultrarapid analysis of NGS data to identify cancer-associated mutations. DRAGEN calls SNVs and indels from both matched tumor-normal pairs and tumor-only samples.

For the tumor-normal pipeline, both samples are analyzed jointly such that germline variants are excluded, generating an output specific to tumor mutations. The tumor-only pipeline produces a VCF file that can be further analyzed to identify tumor mutations. Both pipelines make no ploidy assumptions, enabling detection of low-frequency alleles.

The output, after multiple filtering steps, is in the form of a VCF file. Variants that fail the filtering steps are kept in the output VCF, with a FILTER annotation indicating which filtering steps have failed.

Somatic Mode Options

Somatic mode has the following command line options:

- *--tumor-fastq1* and *--tumor-fastq2*

  The *--tumor-fastq1* and *--tumor-fastq2* options are used to input a pair of FASTQ files into the mapper aligner and somatic variant caller. These options can be used with OTHER FASTQ options to run in tumor-normal mode. For example:

  ```
  dragen -f -r /staging/human/reference/hg19/hg19.fa.k_21.f_16.m_149
     --tumor-fastq1 <TUMOR_FASTQ1>
     --tumor-fastq2 <TUMOR_FASTQ2> -1 <NORMAL_FASTQ1> \
     -2 <NORMAL_FASTQ2>
     --enable-variant-caller true --vc-sample-name RGSM
     --output-directory /staging/examples/ \
     --output-file-prefix SRA056922_30x_e10_50M
  ```

- *--tumor-fastq-list*

  The *--tumor-fastq-list* option is used to input a list of FASTQ files into the mapper aligner and somatic variant caller. This option can be used with other FASTQ options to run in tumor-normal mode. For example:

  ```
  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
```

- *--tumor-bam-input*

  The *--tumor-bam-input* option is used to input a mapped BAM file into the somatic variant caller. This option can be used with other BAM options to run in tumor-normal mode.
- *--vc-min-tumor-read-qual*

  The *--vc-min-tumor-read-qual* option specifies the minimum read quality (MAPQ) to be considered for variant calling. The default value is 20.

Post Somatic Calling Filtering

The following options are available for post somatic calling filtering:

- *--vc-tlod-filter-threshold*

  Set the TLOD filter threshold. The default is 6.5.
- *--vc-enable-clustered-events-filter*

  Enables the clustered events filter. The default is true.
- *--vc-enable-triallelic-filter*

  Enables the multi-allelic filter. The default is true.

Somatic Mode	Filter ID	Description
Tumor-Only & Tumor-Normal	clustered_events	Clustered events (≥ 3) were observed, in a given active region.
Tumor-Only & Tumor-Normal	t_lod	Variant does not meet likelihood threshold (t_lod < 6.5).
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. i.e. ACTACTACT-> ACTACT. Enabled only when using --vc-enable-gatk-acceleration=true.
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 < 30.
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.
Tumor-Only & Tumor-Normal	read_position	Median of distances between start/end of read and a given locus > 5 (the variant is too close to edge of all the reads)
Tumor-Only & Tumor-Normal	panel_of_normals	Seen in at least one sample in the panel of normal VCF.

Post Somatic Calling Filtering in The Presence of a Matched Normal Control Sample

Somatic Mode	Filter ID	Description
Tumor-Normal	germline_risk	Likelihood of allele being present in normal > 0.025.
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 (normalAlleleFraction < (0.1 * tumorAlleleFraction).

gVCF, Combine gVCF, and Joint VCF

Joint calling is a mode in which an individual's genotype sensitivity and specificity can be improved by using information from a related cohort.

gVCF, combine gVCF, and joint calling are implemented in a multistep approach First, a gVCF file is created for each individual. The gVCF is an enriched VCF that contains information not only at the positions where variants were detected, but also at positions that are homozygous reference. However, the gVCF may have discontinuity, ie, there may be gaps between regions reported in the gVCF. Additional information is available that indicates how well the evidence (reads) supports the absence of variants (reference) or alternative alleles.

The DRAGEN gVCF is not necessarily continuous, ie, it may contain gaps, which correspond to regions that are non-callable, and those regions are omitted from the gVCF output.

A region is non-callable if it does not meet the following criterion:

- At least 1 read is mapped with MAPQ>0.

Examples of non-callable regions:

- No reads are mapped to the region.
- More than 1 read are mapped to the regions, but all reads have MAPQ=0.

DRAGEN identifies callable regions upstream from the variant caller, so any gaps in the callable regions remain a gap in the gVCF because the non-callable regions are not passed down through the VC.

In joint calling, if a position is in a gVCF gap for some samples, and in a homref block for other samples, then the position in the joint gVCF is reported as a homref block, but the samples with gaps have their format field as "./.:.:.:." all dots, ie, no call. That position is not reported in the joint VCF because there are no variant calls for any of the individuals.

The following is an example of a position from the joint VCF where one individual has a variant and the other two samples are in a gVCF gap at that same position (their FORMAT field is all dots):

```
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
```

For the next step, you can choose one of the following options:

- Combine gVCF, which takes multiple gVCF files as input and produces one combined gVCF file that represents all the input samples. The combine gVCF file can then be sent to the joint caller.
- Run joint caller directly with multiple gVCF files as input. The gVCF files are jointly genotyped to generate a single VCF. The joint VCF contains multiple genotype columns, each corresponding to a sample in the cohort.

gVCF Options

In addition to the standard parameters for the variant caller stage of the DRAGEN host software, the following parameters are available for gVCF generation:

- *--vc-emit-ref-confidence*

  To enable base pair resolution gVCF generation, set the *--vc-emit-ref-confidence* option to BP_ RESOLUTION. To enable banded gVCF generation, set to GVCF.
- *--vc-gvcf-gq-bands*

  The *--vc-gvcf-gq-bands* option is optional and is used to define GQ bands for gVCF output. The default value is 10,20,30,40,60,80.
- *--vc-max-alternate-alleles*

  The *--vc-max-alternate-alleles* option specifies the maximum number of ALT alleles that are output in a VCF or gVCF. The default value is 6.

Combine gVCF Options

Combine gVCF can be called after the gVCF files are created. The Combine gVCF tool is used to merge input gVCF files to generate a single gVCF file that represents all the input gVCF files.

The following options are required by DRAGEN host software for combine gVCF.

- *--enable-combinegvcfs*

  To enable a combine gVCF run, set to true.
- *--output-directory*

  Specifies the output directory.
- *--output-file-prefix*

  Specifies the prefix used to label all output files for the run.
- *-r*

  Specifies the directory in which the hash table resides.
- *--variant*, *--variant-list*

  Specifies the path to a single gVCF file. Multiple *--variant* options can be used on the command line, one for each gVCF. Up to 500 gVCFs are supported. The *--variant-list* option can be used to provide the path to a file that contains a list of input gVCF files that need to be combined, one variant file path per line.

Joint Calling

Joint calling can be called after the required gVCF files have been created. The following options are required by the DRAGEN host software during joint calling.

- *--enable-joint-genotyping*

  To enable a combine gVCF run, set to true.

- *--output-directory*

  Specifies the output directory.
- *--output-file-prefix*

  Specifies the prefix used to label all output files for the run.
- *-r*

  Specifies the directory in which the hash table resides.
- *--variant*, or *--variant-list*

  Specifies the path to a single gVCF file. Multiple *--variant* options can be used on the command line, one for each gVCF. Up to 500 gVCFs are supported. You can use the *--variant-list* option to specify a file that contains a list of input gVCF files that need to be combined, one variant file path per line.

The following options are optional:

- *--pedigree-file*

  The *--pedigree-file* option specifies the path to a PED pedigree file containing a structured description of the familial relationships between samples. Using this option allows the joint caller to incorporate pedigree information in the analysis. Currently, only pedigree files that contain trios (mother, father, child) are supported.
- *--enable-multi-sample-gvcf*

  To have the joint caller generate a multisample gVCF file, set *--enable-multi-sample-gvcf* to true. This option requires a combined gVCF file as input.

Table 2 Joint-calling Modes, Input Files, and Command-line Options

File to Generate	population joint-called multi-sample gVCF	family joint-called multi-sample gVCF	population joint-called multi-sample VCF	family joint-called multi-sample VCF
Input File	multi-sample combined gVCF file	multi-sample 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 Options	--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

In the joint VCF output records, samples that have FORMAT GT=”0/0” do not have a variant at that position, and therefore may be banded together with other adjacent positions that are also homozygous reference (or have too little evidence for a variant to be called).

For those positions that are banded, the FORMAT/AD, FORMAT/AF, and FORMAT/DP have special meaning, as follows:

- The FORMAT/DP reported for the band is computed as the minimum DP across all positions within the band.
- The FORMAT/AD values reported in the band are the AD values selected at the position in the band where DP = median DP.
- The FORMAT/AF is computed based on FORMAT/AD.

For example:

```
chr1      10230   .             AC          A            66.08    PASS
 AC=2;AF=0.333;AN=6;DP=767;FS=2.949;MQ=30.17;MQRankSum=-
0.808;QD=0.19;ReadPosRankSum=-1.942;SOR=0.866
GT:AD:AF:DP:GQ:FT:F1R2:F2R1:PL:GL:GP
0/1:108,65:0.376:173:19:PASS:63,35:45,30:50,0,45:-5.006,0,-4.51:1.911e+01,5.366e-
02,4.815e+01
0/1:114,70:0.380:184:18:PASS:68,35:46,35:48,0,45:-4.825,0,-
4.486:1.831e+01,6.462e-02,4.792e+01
0/0:223,74:0.249:297:0:LowGQ:.:.:0,0,3318:.:.
```

DeNovo Joint Calling

When a pedigree file is used on the DRAGEN joint calling command line (*--pedigree-file file.ped*), indicating the familial relationships between samples, DRAGEN identifies all variants in the Proband that are in Mendelian conflict and computes an associated DeNovo Quality (DQ) score, which corresponds to the probability of the variant being a DeNovo mutation. The higher the score value, the more likely the variant is DeNovo.

The DeNovo information is output in a slightly different format between the prefilter joint VCF and the postfilter joint VCF.

Prefilter Joint VCF

In the prefilter joint VCF, all variants in the Proband that are in Mendelian conflict are marked with FORMAT/DN as DeNovo, and all have a FORMAT/DQ score printed. All variants in the Proband that are consistent with inheritance from parents are marked with FORMAT/DN as Inherited.

The FORMAT/FT field is also printed as PASS or other annotation, depending on whether the single-sample variant is PASS or not. The FILTER status is marked as '.'.

The following examples show a variant record taken out of the joint prefilter VCF, when called with a pedigree file, with the proband variant marked as DeNovo. In this case, both parents are high confidence homozygous reference, and the child is heterozygous ALT. FORMAT order is proband/father/mother.

```
chr1     861154  .       T       C       27.62   .
       AC=1;AF=0.167;AN=6;DP=61;FS=0.000;MQ=41.44;MQRankSum=-
0.617;QD=0.84;ReadPosRankSum=3.574;SOR=0.850
       GT:AD:AF:DP:GQ:FT:F1R2:F2R1:PL:GL:GP:PP:DQ:DN
       0/1:21,12:0.364:33:31:PASS:11,5:10,7:66,0,45:-6.635,0,-4.45:3.159e+01,3.092e-
03,4.750e+01:24,0,83:6.8575e-04:DeNovo
       0/0:13,0:0.000:11:30:PASS:.:.:0,30,450:.:.:0,26,253
       0/0:16,1:0.059:17:6:PASS:.:.:0,6,592:.:.:22,0,254


chr1     234710899       .       T       C       44.74   .

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
```

Postfilter Joint VCF

In the postfilter joint VCF, all variants in the Proband that are in Mendelian conflict have a FORMAT/DQ score printed, and those that have FORMAT/DQ>= DQ threshold are marked with FORMAT/DN as DeNovo. Those that have FORMAT/DQ<DQ threshold are marked with FORMAT/DN as LowDQ. All variants in the Proband that are consistent with inheritance from parents are marked with FORMAT/DN as Inherited.

The FORMAT/FT field is also printed as PASS or other annotation, depending on whether the single-sample variant is PASS or not. The FILTER status is marked as PASS or other annotation depending on whether the joint QUAL is > a threshold or not.

Taking the same examples given for the prefilter joint VCF, the following are the corresponding records taken out of the joint postfilter VCF. In this case, the first record is no longer marked as DeNovo, but is marked as LowDQ, because FORMAT/DQ is below the threshold. FORMAT order is proband/father/mother.

```
chr1    861154  .       T       C       27.62   DRAGENHardQUAL
        AC=1;AF=0.167;AN=6;DP=61;FS=0.000;MQ=41.44;MQRankSum=-
0.617;QD=0.84;ReadPosRankSum=3.574;SOR=0.850
        GT:AD:AF:DP:GQ:FT:F1R2:F2R1:PL:GL:GP:PP:DQ:DN
        0/1:21,12:0.364:33:31:PASS:11,5:10,7:66,0,45:-6.635,0,-4.45:3.159e+01,3.092e-
03,4.750e+01:24,0,83:6.8575e-04:LOWDQ
        0/0:13,0:0.000:11:30:PASS:.:.:0,30,450:.:.:0,26,253
        0/0:16,1:0.059:17:6:PASS:.:.:0,6,592:.:.:22,0,254


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
```

DeNovo Metrics

In the VC metrics report, the number of DeNovo variants (SNP and INDEL) is counted and reported, for all DeNovo variants above a default DQ threshold. You can specify these thresholds by using the following options:

- *--qc-snp-DeNovo-quality-threshold* <value>, default is 0.05
- *--qc-indel-DeNovo-quality-threshold* <value>, default is 0.02

These thresholds affect the DeNovo counts reported in the VC metrics, as well as the content of the joint postfilter VCF VCF (only those variants in the Proband that are in Mendelian conflict with FORMAT/DQ> DQ threshold are marked with FORMAT/DN as DeNovo). The default threshold values were chosen to achieve a good tradeoff between specificity and sensitivity but can be modified as needed.

In the scenario where there are multiple trios specified in the pedigree file (eg, multi-generation pedigree), DRAGEN automatically detects the trios and assesses the DeNovo variants on the proband sample of each trio.

Germline Variant 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 Variant Hard Filtering

The default filters in the germline pipeline are as follows:

- ##FILTER=<ID=DRAGENHardQUAL,Description="Set if true:QUAL < 10.4139">
- ##FILTER=<ID=PloidyConflict,Description="Genotype call from variant caller not consistent with chromosome ploidy">
- ##FILTER=<ID=lod_fstar,Description="Variant does not meet likelihood threshold (default threshold is 6.3)">
- DRAGENHardQUAL: For all contigs other than the mitochondrial contig, the default hard filtering consists of thresholding the QUAL value only.
- lod_fstar: For the mitochondrial contig, the default hard filtering consists of thresholding the LOD score only.
- PloidyConflict: This filter is applied to all variant calls on chrY of a female subject, if female is specified on the DRAGEN command line.

Non-Default 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 <, ≤, =, ≠, ≥, 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.

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.

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.

Panel of Normals VCF

When DRAGEN is used in the Tumor-Normal or Tumor-Only somatic mode, it looks up variant calls in a panel of normals (PON) VCF. The PON VCF must be generated ahead of time and represents a set of variants that were detected by the DRAGEN Somatic Pipeline when run on a set of normal samples (which are not necessarily matched to the subject from whom the tumor sample was taken). Ideally though, the PON VCF should come from normal samples collected on the same library prep/sequencing instrument, so that if there are systematic errors that occur during sequencing/library prep, they get captured in the PON VCF.

- *--panel-of-normals*

  Specifies a PON VCF file. When a PON VCF file is used as input, if a somatic variant is found in at least one sample in the file (which may contain several dozen samples), it is marked as panel_of_ normals in the FILTER column of the output VCF.

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.

Copy Number Variant Calling

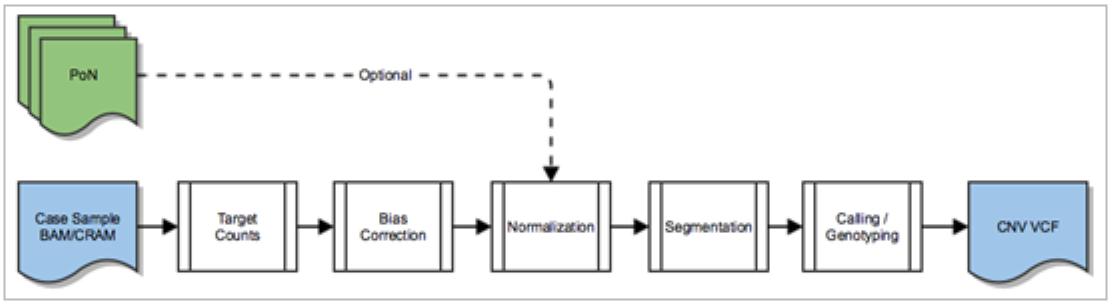
The DRAGEN Copy Number Variant (CNV) pipeline can call CNV events using next-generation sequencing (NGS) data. This pipeline supports multiple applications in a single interface via the DRAGEN Host Software, including processing of whole-genome sequencing data and whole-exome sequencing data for germline 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 3 DRAGEN CNV Pipeline Workflow



This 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, which extracts signals such as read count and improper pairs and puts them into target intervals.

Figure 4 Read Count Signal

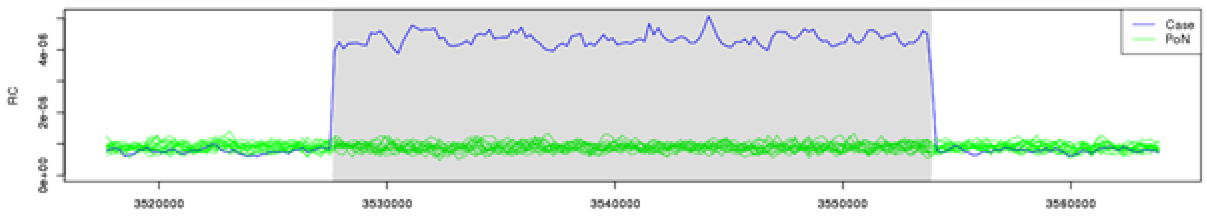
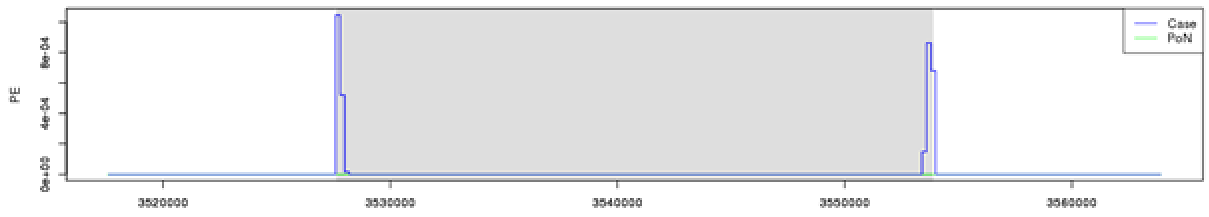
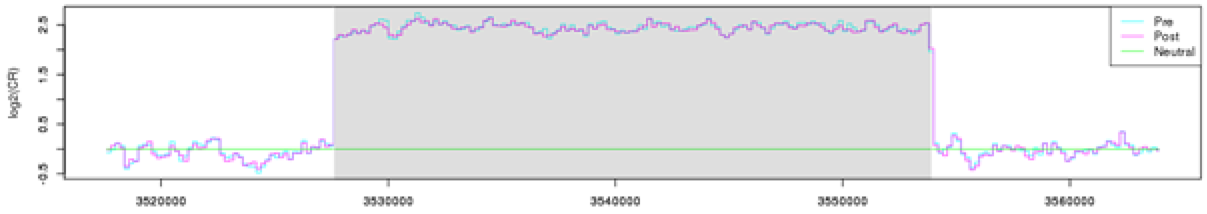


Figure 5 Improper Pairs Signal



Next, the case sample is normalized against the panel of normals, or against the estimated normal ploidy level, and any other biases are subtracted out of the signal to amplify any event level signals.

Figure 6 Pre/Post Tangent Normalization



The normalized signal is then segmented using one of the available segmentation algorithms, and events are called from the segments.

Figure 7 Segments

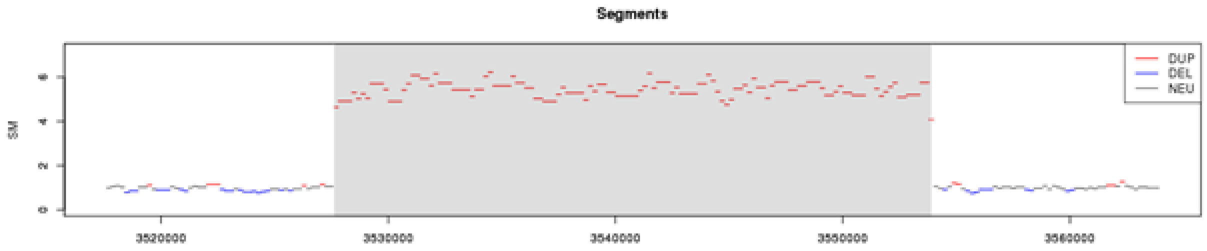
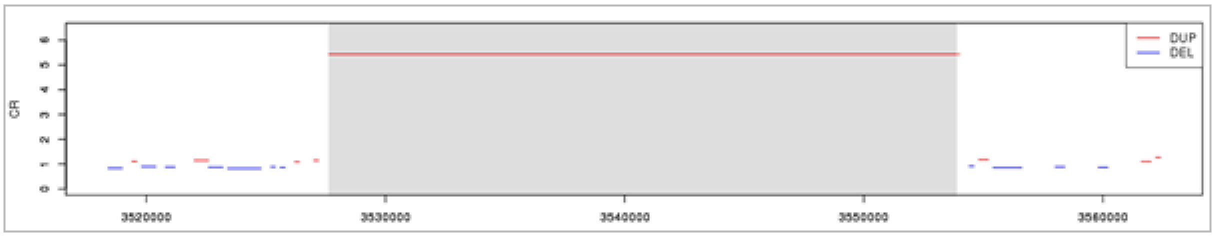


Figure 8 Called Events



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. The input into the DRAGEN CNV can be a BAM or CRAM file. If you are using the DRAGEN mapper and aligner, FASTQ files can be used.

- *--bam-input*–The BAM file to be processed.
- *--cram-input*–The CRAM file to be processed.
- *--enable-cnv*–Enable or disable CNV processing. Set to true to enable CNV processing.
- *--enable-map-align*–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.
- *--fastq-file1*, *--fastq-file2*–FASTQ file, or files, to be processed.
- *--output-directory*–Output directory where all results are stored.
- *--output-file-prefix*–Output file prefix that will be prepended to all result file names.
- *--ref-dir*–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 43.

To run the DRAGEN CNV Pipeline directly with FASTQ input without generating a BAM or CRAM file, then see *Streaming Alignments* on page 44, which outlines steps for 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 kmer uniqueness map that the CNV algorithm uses to counteract mapability biases. The kmer uniqueness map file only needs to be generated once per reference hashtable and 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 *Preparing a Reference Genome* on page 100.

The following is an example of a command to generate a hashtable.

```
dragen \
   --build-hash-table true \
   --ht-reference $FASTA \
   --output-directory $OUTPUT \
   --enable-cnv true \
   --enable-rna true
```

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 DRAGEN CNV Pipeline.

You need to generate alignment files for all samples that have not already been mapped and aligned, including any samples to be used as references for normalization. Each sample must have a unique sample identifier, which is specified with the *--RGSM* option. For BAM and CRAM input files, the sample identifier is taken from the file, so the *--RGSM* option is not required.

Example command to map/align 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
```

Example command to map/align an existing BAM file:

```
dragen \
   -r $HASHTABLE \
   --bam-input $BAM \
   --output-directory $OUTPUT \
   --output-file-prefix $SAMPLE \
   --enable-map-align true
```

Example command to map/align 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. Examples of downstream callers include the CNV Caller and the Haplotype Variant Caller. This process allows you to skip generation of a BAM or CRAM file, bypassing 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 shows an example command line to map/align a FASTQ file and send it to the CNV 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 Haplotype Variant Calling* on page 55.

Target Counts

The target counts stage is the first processing stage for the DRAGEN CNV pipeline. This step 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-wgs-skip-contig-list* option.

With whole exome sequence data, the target BED file supplied with the *--cnv-target-bed* option is used to determine the intervals for analysis.

The targets counts stage generates a .target.counts file, which can be later used in place of any BAM or CRAM by specifying it with the *--cnv-input* option for the normalization stage. The .target.counts file is an intermediate file for the DRAGEN CNV pipeline and should not be modified.

The .target.counts file is a tab-delimited 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

An example of a *.target.counts file is shown below.

```
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-wgs-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 normal because all the samples must have matching intervals. For self normalization, the effective width may be larger than the specified value.

WGS Coverage	Recommended Resolution*
5x	1000 bp
10x	1000 bp
20x	500 bp
30x	250 bp
50x	250 bp

*You can choose to trade off between resolution and speed.

The intervals are autogenerated for every contig in the reference. You can specify a list of contigs to skip by using the *--cnv-wgs-skip-contig-list* option. This option takes comma-separateded 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-wgs-skip-contig-list "chrM,chrX,chrY"
```

Whole Exome

If the samples are whole exome samples, a target BED file should be supplied with the *--cnv-target-bed $TARGET_BED* option.

The target BED file requires a header line and a fourth column, which indicates the target name. The following is a simple awk command that translates an input target BED file (without any headers or comment lines) to one that is suitable for CNV.

```
cat <(echo -e "contig\tstart\tstop\tname") \
<(awk '{print $1"\t"$2"\t"$3"\ttarget-"NR}' $ORIGINAL_BED)
```

A regular BED file without a header is also allowed, in which case the fourth column target names are auto generated by the CNV algorithm during the target counts stage. 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.

- *--cnv-counts-method*—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 normal approach, which means if an alignment overlaps any part of the target bin, it is counted for that bin. In the self normalization mode, the default counting method is start.
- *--cnv-enable-split-intervals*—When this option is set to true, splits up all target BED intervals into two equally spaced intervals. When this option is enabled, then all samples (case and panel of normals) must be executed with this option enabled. The default is false.
- *--cnv-min-mapq*—Specifies the minimum MAPQ for a read to be counted during target counts generation. Note that MAPQ 0 reads are always counted, independent of this setting. The default value is 20.
- *--cnv-target-bed*—Specifies a properly formatted BED file that indicates the target intervals to sample coverage over. For use in WES analysis.
- *--cnv-wgs-interval-width*—Specifies the width of the sampling interval for CNV WGS processing. This option controls the effective window size. The default is 1000.
- *--cnv-wgs-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".

GC Bias Correction

Biases are introduced in a typical NGS workflow, which make calling CNV events difficult. These biases can arise from library prep, capture kits, sequencer differences, and even mapping biases. The DRAGEN CNV pipeline attempts to correct for these biases in the various processing stages.

The GC bias correction module immediately follows the target counts stage and operates on the .target.count file. GC bias correction generates a GC bias corrected version of the file, which has a .target.counts.gc-corrected 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 may be skewed), then GC bias correction should be disabled.

The following options control the GC bias correction module.

- *--cnv-enable-gcbias-correction*—Enable or disable GC bias correction when performing target counts generation. 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—Uses statistics from the sample under analysis to determine the baseline ploidy levels.
- 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 in this case 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

Panel of Normals

- Whole genome sequencing
- Whole exome sequencing
- Targeted panels
- Additional matched samples are available
- Tumor/Matched-Normal analysis

Self Normalization

The DRAGEN CNV pipeline provides the self normalization mode that does not require a reference sample or a panel of normals. Enable this mode by setting *--cnv-enable-self-normalization* to true. This operating mode bypasses the need to run two stages and can save time. It uses the statistics within the case sample itself 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, producing 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-wgs-interval-width* option that is used during the target counts stage becomes the effective interval width based on the number of unique kmer positions. You typically do not have to modify this option.

Panel of Normals

The Panel of Normals approach 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 out system level biases that are not sample specific.

In this mode of operation, 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

Target counts should be performed for all your samples, whether they are to be used as references or are the case samples under investigation. The case sample 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, which is enabled by default.

The examples below are for WGS processing. For exome processing, see *Whole Exome* on page 46.

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 \
   "
```

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 \
   "
```

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, keeping in mind that 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, it is recommended that you use sex matched references in the panel. Because the normalization is performed on a per-target basis against the panel of normals median, having sex matched references is critical to detecting copy number events on the sex chromosomes.

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 file generated from the target counts stage. Relative paths are supported provided 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, in which uses a subset of the GC corrected files from the target counts stage.

```
/data/output_trio1/sample1.target.counts.gc-corrected
/data/output_trio1/sample2.target.counts.gc-corrected
/data/output_trio2/sample4.target.counts.gc-corrected
/data/output_trio2/sample5.target.counts.gc-corrected
/data/output_trio3/sample7.target.counts.gc-corrected
/data/output_trio3/sample8.target.counts.gc-corrected
```

Alternatively, the files to be used in the panel of normals can be specified with the *--cnv-normals-file* option. This option takes 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. Because we recommend using the GC bias corrected counts from the previous stage, there is no need to run GC bias correction again.
GC bias correction can be disabled by setting *--cnv-enable-gcbias-correction* to false. For example:

```
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 \
"
```

This command normalizes the case sample against the panel of normals and then continues downstream to the segmentation and calling stage.

Normalization Options

These options control the preconditioning of the panel of normals and the normalization of the case sample.

- *--cnv-enable-self-normalization*—Enable/disable self normalization mode, which does not require a panel of normals.
- *--cnv-extreme-percentile*—Specifies the extreme median percentile value at which to filter out samples. The default is 2.5.
- *--cnv-input*—Specifies a target counts file for the case sample under investigation when using a panel of normals.
- *--cnv-matched-normal*—Specifies the target counts file of the matched normal sample.
- *--cnv-normals-file*—Specifies a target.counts file to be used in the panel of normals. This option can be specified multiple times, once for each file.
- *--cnv-normals-list*—Specifies a text file containing 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 provided the paths are relative to the current working directory.
- *--cnv-max-percent-zero-samples*—Specifies a threshold for filtering out targets with too many zero coverage samples. The default is 5%.
- *--cnv-max-percent-zero-targets*—Specifies a threshold for filtering out samples with too many zero coverage targets. The default is 2.5%.
- *--cnv-target-factor-threshold*—Specifies a percentage of median target factor threshold to filter out useable targets. The default is 1% for whole genome processing and 10% for targeted sequencing processing.
- *--cnv-truncate-threshold*—Specifies a percentage threshold for truncating extreme outliers. The default is 0.1%.

Segmentation

After a case sample has been normalized, it goes through a segmentation stage. There are multiple segmentation algorithms implemented in DRAGEN, including the following:

- CBS (Circular Binary Segmentation)
- SLM (Shifting Level Models)
- FPOP (Functional Pruning Optimal Partitioning)

The SLM algorithm has two variants, SLM and HSLM. HSLM (Heterogeneous SLM) is for use in exome analysis and handles target capture kits that are not equally spaced.

The default segmentation algorithm in use is SLM for whole genome processing, and CBS for whole exome processing.

- *--cnv-segmentation-mode*—Specifies the segmentation algorithm to perform. Values are cbs, slm, hslm, or fpop. The default value is slm or cbs depending on whether the intervals are whole genome intervals or targeted sequencing intervals.
- *--cnv-merge-distance*—Specifies the minimum number of base pairs between two segments that would allow them to be merged. The default value is 0, which means the segments must be directly adjacent.
- *--cnv-merge-threshold*—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. Set to 0 to disable merging.

External R Packages

To use the SLM or FPOP segmentation algorithms, you must install external R packages. R packages are executed externally outside of the DRAGEN Host Software. A simple installation script is packaged with DRAGEN and can be found at /opt/edico/R/install_R_packages.R. These R Packages are additional segmentation methods that can be used. They are not officially maintained by the DRAGEN Host Software. If your system does not allow the installation of R or third party libraries, then you can default to Circular Binary Segmentation (CBS).

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* with enhancements to improve sensitivity for NGS data.

The following options control Circular Binary Segmentation.

- *--c-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.

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.

- *--cnv-slm-eta*—Baseline probability that the mean process changes its value. The default is 1e-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 modulating relative weight between experimental/biological variance. The default is 0.3.

- *--cnv-slm-stepeta*—Distance normalization parameter. The default is 10000. This option is only valid for HSLM.

Functional Pruning Optimal Partitioning Segmentation

The FPOP segmentation mode follows from the R implementation of On Optimal Multiple Changepoint Algorithms for Large Data.

- *--cnv-fpop-penalty*—Penalty option for changepoint detection. The default is 0.03.

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-scale metric measuring 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.

Output Files

The DRAGEN host software generates many intermediate files. The final call file that contains the amplification and deletion events is the *.seg.called.merged file.

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 have copy neutral (REF) calls included in the output VCF, set *--cnv-enable-ref-calls* to true.

For more information about the .seg.called.merged file, and how to use the output files to aid in debug and analysis, see *Signal Flow Analysis* on page 41.

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 it 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=VCFv4.1
##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">
##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
##INFO=<ID=REFLEN,Number=1,Type=Integer,Description="Number of REF positions
```

```
included in this record">
##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=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">
```

The ID column is used to represent 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 it contains the name of the failed filter.

The INFO column contains information representing the event, mostly identical to the ID column. 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. The CIPOS and CIEND entries are currently not used.

The FORMAT fields are described in the header.

- 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

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.

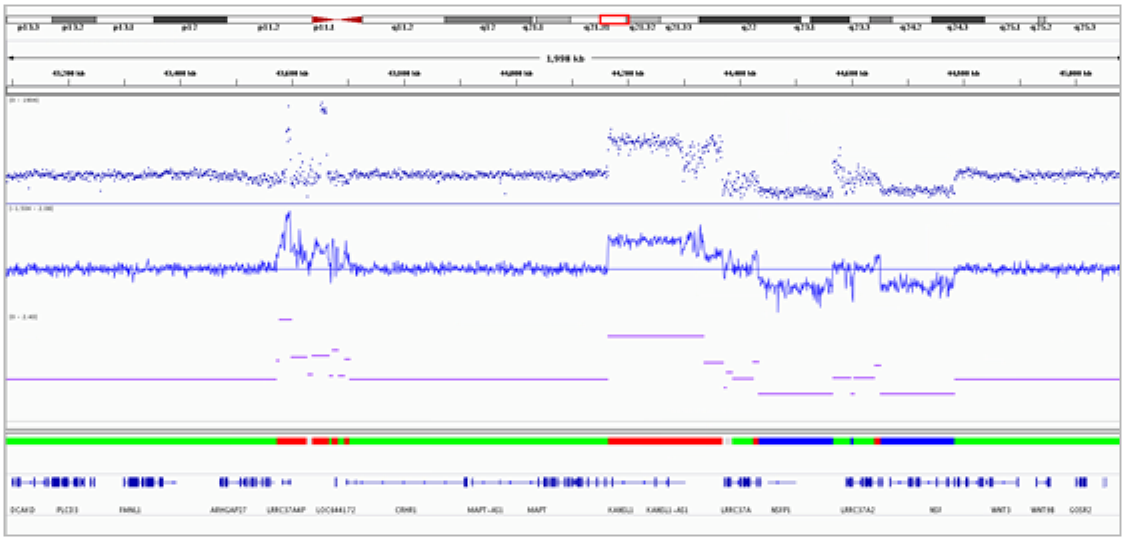
All files have a structure similar to a BED file, with an optional header line.

- **.target.counts*—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 normals samples. There is also a bigwig representation of a target.counts.diploid file, which is normalized to the normal ploidy level of 2 instead of raw counts.
- **.tn.tsv*—The case sample's tangent normalized signal, per target interval. This file contains the log-normalized signal. A strong signal deviation from 0.0 indicates a potential for a CNV event.

- **.seg.called.merged*—Contains the segments produced from the segmentation algorithm.
- **.cnv.vcf*—Output CNV VCF file indicating 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. An example is shown in the following figure.

Figure 9 IGV Example



Output and Filtering Options

The output and filtering options control the CNV output files.

- *--cnv-blacklist-bed*—Specifies a BED file indicating intervals to exclude from the final output CNV VCF. If a call overlaps any interval from the blacklist BED file by at least 50%, it is suppressed.
- *--cnv-enable-plots*—Generate plots as part of the CNV pipeline. The default is false.
- If you perform WGS CNV analysis with high-resolution intervals (less than 1000 bp), then plot generation can take longer to complete. Illumina recommends that you use the default (disabled).
- *--cnv-enable-ref-calls*—Emit copy neutral (REF) calls in the output VCF file. The default is false.
- *--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. 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 it will be 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, leading 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. The default is 10.
- *--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-ploidy*—Specifies the normal ploidy value. This option is used only for estimation of the copy number value emitted in the output VCF file. The default is 2.
- *--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 Haplotype Variant Calling

DRAGEN can perform mapping and aligning of FASTQ samples and then directly stream the data to downstream callers. A single sample can run through both the CNV and the Haplotype VC if the input is a FASTQ sample. This triggers self normalization by default.

Run the FASTQ sample through a regular DRAGEN map/align workflow, and then provide additional arguments to either enable the CNV or 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 w/ 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 \
   --vc-sample-name $SAMPLE
```

Map/Align FASTQ w/ 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 \
   --vc-sample-name $SAMPLE \
   "
```

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 \
   --vc-sample-name $SAMPLE
```

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
```

To perform analysis on the sex chromosomes using a panel of normals, it is recommended that you use sex matched samples in the panel of normals.

You can override the sex of the sample with the *-sample-sex* option .

Multisample CNV Calling

Multisample CNV calling is possible starting from tangent normalized counts files (*.tn.tsv) 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.

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
```

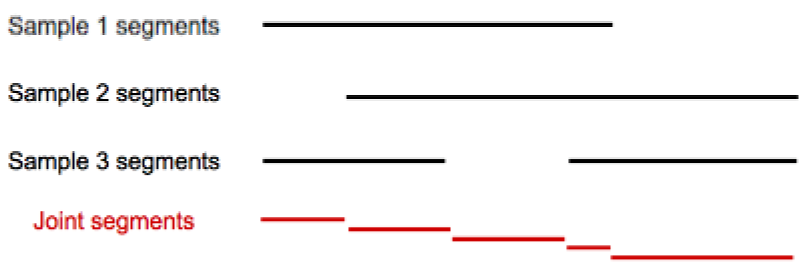
De Novo CNV Calling Options

The following options are used in DeNovo CNV calling:

- *--cnv-input*–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.
- *--cnv-filter-de-novo-qual*–Phred-scaled threshold at which a putative event in the proband sample is marked as DeNovo. Default value is 0.10.
- *--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. For example:



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 mutlisample 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 mutlisample 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

Mother Copy Number	Father Copy Number	Proband Copy Number	Mendelian Consistent?
3	2	4	No
3	2	2	Yes

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 0.1)
- DeNovo - the called trio genotype is inconsistent with Mendelian inheritance and DQ is greater than or equal to the de novo quality threshold (default 0.1)

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 > 0.100000">
```

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.

Repeat Expansion Detection with Expansion Hunter

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 thereby 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. This method works by performing an accurate sequence-graph based realignment of reads that originate inside and around each target repeat. It then genotypes the length of the repeat in each allele based on these graph alignments. More information and analysis is available in the following ExpansionHunter papers:

- ExpansionHunter (original version)
- Graph ExpansionHunter (new version integrated in DRAGEN)

It is important to note that these methods work only for whole human genome samples generated with PCR-free methods. Repeats are only genotyped if the coverage at the locus is at least 10x.

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 spec file>

In addition the sex of the sample should be set using the *--sample-sex* option.

The following options are optional.

- --repeat-genotype-region-extension-length=<length of region around repeat to examine> (default 1000bp)
- --repeat-genotype-min-baseq=<Minimum base quality for 'high confidence' bases> (default 20)

For more information on the spec file specified by *--repeat-genotype-specs* option, see *Repeat Expansion Specification Files* on page 60.

The main output of repeat expansion detection is a VCF file, containing 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 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 Expansion Detection Output Files

VCF Output File

The results of repeat genotyping are output as a separate VCF file, giving 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 begins with the following fields.

Table 3 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> where n is the number of repeat units
QUAL	Always .
FILTER	Always PASS

Table 4 Additional INFO Fields

Field	Description
SVTYPE	Always STR
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
RU	Repeat unit in the reference orientation
REPID	Repeat id from the repeat-specification file

Table 5 GENOTYPE (Per Sample) Fields

Field	Description
GT	Genotype
SO	Type of reads that support the allele; can be SPANNING, FLANKING, or INREPEAT meaning that the reads span, flank, or are fully contained in the repeat
CI	Confidence interval called repeat length of each allele
AD_SP	Number of spanning reads consistent with the allele
AD_FL	Number of flanking reads consistent with the allele
AD_IR	Number of in-repeat reads consistent with the allele

For example, the following VCF entry describes the state of C9orf72 repeat in a sample with ID LP6005616-DNA_A03.

```
QUAL     FILTER   INFO     FORMAT   LP6005616-DNA_A03
chr9     27573526          .        C        <STR2>,<STR349> .        PASS
SVTYPE=STR;END=27573544;REF=3;RL=18;RU=GGCCCC;REPID=ALS GT:SO:CN:CI:AD_SP:AD_
```

```
FL:AD_IR
1/2:SPANNING/INREPEAT:2/349:2-2/323-376:19/0:3/6:0/459
```

In this example, the first allele spans 2 repeat units while the second allele spans 349 repeat units. The repeat unit is GGCCCC (RU INFO field), so the sequence of the first allele is GGCCCCGGCCCC and the sequence of the second allele is GGCCCC x 349. The repeat spans three 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 (323,376). There are 19 spanning and 3 flanking reads consistent with the repeat allele of size 2 (that is 19 reads fully contain the repeat of size 2 and 2 flanking reads overlap at most 2 repeat units). Also, there are 6 flanking and 459 in-repeat reads consistent with the repeat allele of size 349.

Additional Output Files

In addition to the main VCF output, the repeat genotyper also outputs the same information in a JSON file for easy parsing.

The new sequence-graph alignments of reads in the targeted repeat regions are output in a BAM file. The alignment position of each read is set to the position in the repeat-region to which it realigned. The full alignment (CIGAR) against the sequence graph is given in the custom XG tag in the format <LocusName>,<StartPosition>,<GraphCIGAR>

StartPosition is the first alignment position of the read in the first node and GraphCIGAR describes the alignment against the graph from there. For reference see https://git.illumina.com/Bioinformatics/graph-tools/blob/master/docs/alignment.md. Quality scores in this BAM file are transformed into binary representation, one high score (40) and one low score (0), used by graph-genotyping.

SMA Calling

Disruption of all copies of the SMN1 gene in an individual causes spinal muscular atrophy (SMA), a progressive disease affecting ~1 in 10,000 live births. SMN1 has a very high identity paralog, SMN2, with 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 uses sequence-graph realignment (also used for repeat expansion calling) to align reads to a single reference representing SMN1 and SMN2. In addition to the standard diploid genotype call, the program 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.

SMA calling is only supported for human whole-genome sequencing samples with PCR-free libraries.

Usage

SMA calling is implemented together with repeat expansion detection. For information on graph-alignment and options, see *Repeat Expansion Detection with Expansion Hunter* on page 60.

SMA calling is enabled, along with repeat expansion detection, by setting the *--repeat-genotype-enable* option to true. To activate SMA calling, the variant specification catalog file must include a description of the targeted SMN1/2 variant. Example files are available in the /opt/edico/repeat-specs/experimental folder.

SMN output is included along with any targeted repeats in <outputPrefix>.repeat.vcf. SMN output is represented as a single SNV call at the key (splice-affecting) position in SMN1, with SMA status in custom fields:

Table 6 SMA Result in repeat.vcf Output

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 affected, - indicates unaffected, ? indicates uncertain,
AD	Total read counts supporting the C and T allele.
RPL	Log10 Likelihood ratio between the affected and unaffected models. Positive scores are in favor of unaffected.

Structural Variant Calling

DRAGEN has integrated the methods of the Manta Structural Variant Caller based on version 1.5.1. For descriptions of these methods, see Manta Documentation.

Structural variants (SVs) and indels are called from mapped paired-end sequencing reads. The SV caller is optimized for analysis of germline variation in small sets of individuals and somatic variation in tumor-normal sample pairs.

The SV caller does the following:

- 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.
- Provides scoring models for germline variants in small sets of diploid samples and somatic variants in matched tumor-normal sample pairs.

There is experimental support for analysis of unmatched tumor samples as well. All SV and indel inferences are output in VCF 4.1 format.

Structural Variant Calling Options

The following command line options are supported for the Structural Variant Caller:

- *--enable-sv*—Enable or disable the structural variant caller. The default is false.
- *--sv-reference*—Specifies a reference file in FASTA format.
- *--sv-call-regions-bed*—Specifies a BED file containing the set of regions to call. The file can be unzipped text or Bgzip-compressed/tabix-indexed.
- *--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".

- *--sv-exome*–When set to true, sets options for WES input, which includes disabling depth filters. 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-quiet*–Set to true to suppress log output to stderr (still writes to log file). The default is true.

Modes of Operation

Structural Variant calling can run in the following modes:

- Standalone–Uses mapped BAM/CRAM input files. 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*

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

The following is an example command line for Integrated mode:

```
dragen -f
   --sv-reference=<FASTA> \
   --ref-dir=<HASH_TABLE> \
   --enable-map-align=true \
   --enable-map-align-output=true \
   --output-format=BAM \
   --enable-sv=true \
   --output-directory=<OUT_DIR> \
   --output-file-prefix=<PREFIX> \
   -1 <FASTQ1> \
   -2 <FASTQ2>
```

The following is an example command line for Joint Diploid calling in Standalone mode:

```
dragen -f
   --sv-reference=<FASTA> \
   --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>
```

Output Files

The structural variants VCF output file is available in the output directory. The file is named *<output-file-prefix>.sv.vcf.gz*.

The Structural Variant caller also produces also output in the <output-directory>/sv/results and <output-directory>/sv/workspace folders. The results folder contains stats files and variants. The variants are output in a Bgzip-compressed/tabix-indexed VCF file. The workspace folder contains logs and temporary files that are used for debugging. The temporary files are automatically deleted, but can be kept by setting the *--sv-retain-temp-files* option to true.

Structural Variant De Novo Quality Scoring

De novo quality scoring can be enabled for structural variant joint diploid calling, by setting *--sv-denovo-scoring* to true and supplying a pedigree file. This adds a FORMAT/DQ field to the output VCF file.

The following example shows a command line for enabling the de novo quality scoring for a joint diploid run.

```
dragen -f
   --sv-reference=<FASTA> \
   --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>
```

Dragen can also be run on an existing Structural Variant output VCF containing multiple samples (ie, a Trio with a Proband and Parents) to generate a modified VCF file that contains a FORMAT/DQ field (the original file is not changed).

The following example shows a command line for deriving the de novo quality score from an existing SV trio.

```
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>
```

The DQ field is defined as follows:

```
##FORMAT=<ID=DQ,Number=1,Type=Float,Description="Denovo quality">
```

The DQ field represents a score of the posterior probability of the variant being denovo in the proband. If it can be calculated, the score in Phred scale is added to the proband, while the other samples are marked with a period (.) to indicate missing.

The inputs to the function are the VCF file and the Pedigree file that specifies which sample in the trio is the proband, mother, or father.

QC Metrics and Coverage (Enrichment) Reports

Pipeline-specific metrics are generated during each run. The metrics are autogenerated and do not require any activation or specific commands. Metric calculation is performed during analysis so that it does not impact the DRAGEN run time.

Table 7 Metrics Reports

Report name	Equivalent command	Default report?	DRAGEN output
full_res	DRAGEN/samtools genomecov	No	<output prefix>_<coverage region>_full_res.bed
cov_report	DRAGEN's depth of coverage report	No	<output prefix>_<coverage region>_cov_report.bed
hist	DRAGEN's default histogram	Yes	<output prefix>_<coverage region>_hist.txt
mean_cov	Mean coverage over region	Yes	<output prefix>_<coverage region>_mean_cov.txt

Where <coverage region> is one of the following:

- wgs
- target_bed
- qc-coverage-region-1
- qc-coverage-region-2
- qc-coverage-region-3

Mapping and Aligning

Mapping and aligning metrics, like 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).

- **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 used.
- **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 used.
- **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 not passing 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.
- **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.
- **Histogram of reads map qualities**
  - 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.
- **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.
- **Average sequenced coverage over genome/target region**–Number of all reads in Fastqs * read length divided by the number of sites in the target bed or genome.
- **Average alignment coverage over genome/target region**–Number of uniquely mapped bases (ignoring duplicate reads and any clipped bases, only including reads with MAPQ > 0) divided by the number of sites in the target bed or genome.
- **Histogram**–See *Hist Coverage Report* on page 72.
- **Average chromosome X coverage over genome/target region**–Total number of bases that aligned to chromosome X (or to the intersection of chromosome X with the target region) divided by the total number of loci in chromosome X (or to the intersection of chromosome X with the target region). If there is no chromosome X in the reference genome, this metric shows as NA.
- **Average chromosome Y coverage over genome/target region**–Total number of bases that aligned to chromosome Y (or to the intersection of chromosome Y with the target region) divided by the total number of loci in chromosome Y (or in the intersection of chromosome Y with the target region). If there is no chromosome Y in the reference genome, this metric shows as NA.
- **XAvgCov/YAvgCov ratio over genome/target region**–Average chromosome X alignment coverage in the genome (or in the target region) divided by the average chromosome Y alignment coverage in the genome (or in the target region). If there was chromosome X or chromosome Y in the reference genome, this metric shows as NA.

- **Average mitochondrial coverage over genome/target region**—Total number of bases that aligned to the mitochondrial chromosome (or to the intersection of the mitochondrial chromosome with the target region) divided by the total number of loci in the mitochondrial chromosome (or in the intersection of the mitochondrial chromosome with the target region). If there was no mitochondrial chromosome in the reference genome, this metric shows as NA.
- **Average autosomal coverage over genome/target region**—Total number of bases that aligned to autosomes (or to the autosomal loci in the target region) divided by the total number of loci in the autosomes (or to the autosomal loci in the target region). If there was no autosome in the reference genome, this metric shows as NA.
- **Median autosomal coverage over genome/target region**—Median alignment coverage over the autosomes (or over the autosomal loci in the target region). If there was no autosome in the reference genome, this metric shows as NA.
- **Mean/Median autosomal coverage ratio over genome/target region**—Mean autosomal coverage in the genome (or in the target region) divided by the median autosomal coverage in the genome (or in the target region). If there was no autosome in the reference genome, this metric shows as NA.
- **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.
- **Capture specificity**—Number of unique (nonduplicate) reads that mapped inside the target region with a MAP quality greater than 0, divided by the number of unique (nonduplicate) reads that mapped anywhere in the reference with a MAP quality greater than 0.
- **Estimated sample contamination**—When variant calling is conducted with any bioinformatics tool, prediction accuracy is highly 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 pileup locations. The mixture model accounts for the population allele frequencies and the inferred sample genotypes.

  To enable this metric, you must provide the file path on the command line to a VCF that includes marker sites (RSIDs) with population allele frequencies. For example:

  ```
  --qc-cross-cont-vcf /opt/edico/config/sample_cross_contamination_
     resource_hg19.vcf
  ```

  The VCF resource files that are included with DRAGEN can be reconstructed from the Ensambl database. The VCFs included in DRAGEN’s 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, CRCh37 resource file and hg19 reference).

  The following shows example output for a sample with no contamination. This value is provided as a fraction, so a value of 0.011 the same as 1.1% estimated contamination.

  ```
  MAPPING/ALIGNING SUMMARY Estimated sample contamination 0.000
  ```

Variant Calling

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) VCFs.

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, counting the reference as one, and therefore allowing for one variant allele.

Multiallelic–Number of sites in the VCF that contain three or more observed alleles. The reference is counted as one, therefore allowing for two or more variant alleles.

SNPs–A variant is counted as an SNP when the reference, allele 1, and allele2 are all length 1.

INDELs–Every other variant side is considered an Indel, ie, if allele 1 is of length 1, but allele 2 is not, then the variant site is classified as an indel.

MNPs–A variant is counted as a MNP when the alleles 1 and 2 and also the ref sequence are of equal length and longer than 1.

DeNovo SNPs–DeNovo marked SNPs, with DQ > 0.05. Sett the *--qc-snp-denovo-quality-threshold* option to the required threshold. The default is 0.05.

DeNovo INDELs–DeNovo marked indels with DQ values > 0.02. This DQ threshold can be specified by Setting the *--qc-indel-denovo-quality-threshold* option to the required DQ threshold. The defaultis 0.02.

DeNovo MNPs–Same as the option for SNPs. Set the *--qc-snp-denovo-quality-threshold* 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 *--bsnp* option, then both the **In dbSNP** and **Novel** metrics show as NA.

Novel–Total number of variants minus number of variants in dbSNP.

Duration

A breakdown of the run duration for each process, eg, reference loading, aligning reads, and variant calling.

QC Metrics Output Format

The QC metrics are printed to the standard out in a human friendly format and csv files are written to the run output directory.

- <output prefix>.mapping_metrics.csv
- <output prefix>.time_metrics.csv
- <output prefix>.vc_metrics.csv

Each row is self-explained, making it easy to parse.

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
...				
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
...				
RUN TIME		Time loading reference	00:18.6	18.65
RUN TIME		Time aligning reads	19:24.4	1164.42

Coverage/Enrichment Reports

DRAGEN generates a set of default coverage reports for either the whole genome, or, if the *--vc-target-bed* option is specified, for the target region.

In addition to these default reports, the coverage reports feature allows the user to specify up to three regions of interest (coverage regions). For each of these regions, DRAGEN generates the default reports, and any optional report requested for the region.

You can enable the region coverage reports with the *-qc-coverage-region-i* options, where i can equal 1, 2, or 3. Each *-qc-coverage-region-i* option specifies a bed file and has an optional associated *-qc-coverage-reports-i* option, which specifies the type of reports that should be generated for each specific region in addition to the default reports.

All default and optional coverage reports listed in the two tables below use the following default rules for counting reads and bases:

- Duplicate reads are ignored.
- Soft and/or hard-clipped bases are ignored
- Reads with MAPQ=0 are ignored.
- Overlapping mates are double-counted.

Non-default settings:

It is possible to override the min MAPQ and min BQ to apply for a given region using the qc-coverage-filters. For more information, see *MAPQ and BQ Coverage Filtering* on page 74.

The reports are available with or without running either the mapper and aligner, or the variant caller. However, the *--enable-sort* options needs to be set to true (the default is true).

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.

Table 8 Default Reports

Report Name	DRAGEN Output File Type
Hist Coverage	_hist.csv
Overall mean coverage	_overall_mean_cov.csv
Per contig mean coverage	_contig_mean_cov.csv
Predicted ploidy	_ploidy.csv

Table 9 Optional Reports

Report Name	DRAGEN Output File Type
full_res	_full_res.bed
cov_report	_cov_report.bed

Coverage Reports Use Cases and Expected Output

--vc-target-bed specified? Y/N	--qc- coverage-region-i specified? Y/N	Expected Output Files
N	N	• Four .wgs_ default reports
N	Y	• Four .wgs_ default reports • Four coverage region default reports • Up to two optional coverage region reports as specified on the command line

--vc-target-bed specified? Y/N	--qc- coverage-region-i specified? Y/N	Expected Output Files
Y	N	• Four .target_bed_ default reports
Y	Y	• Four .target_bed_ default reports • Four <coverage region> default reports • Up to two optional <coverage region> reports as requested by the user

The following is an example of the additional arguments required to generate coverage reports:

```
$ dragen … \
   --qc-coverage-region-1 <bed file 1> \
   --qc-coverage-report-1 full_res mean_cov \
   --qc-coverage-region-2 <bed file 2> \
   --qc-coverage-region-3 <bed file 3> \
   --qc-coverage-report-3 cov_report mean_cov
```

Hist Coverage Report

The hist report outputs a _hist.csv file, which provides the following:

- Percentage of bases in the genome/target region/coverage 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)", "[50x:100x)", "[20x:50x)", "[10x:20x)", "[3x:10x)", "[0x:
   3x)"
```

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
```

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	Column3
Contig name	Number of bases aligned to that contig, which excludes bases from duplicate marked reads, reads with MAPQ=0, and clipped bases.	Estimated coverage, 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>.

Contig lengths and target bed region lengths (used as denominators) include regions with N in the FASTA.

Ploidy Report

The ploidy report generates a predicted ploidy in a _whg_ploidy.txt file.

The following is an example of the contents of the _ploidy.txt file:

```
Predicted sex chromosome ploidy XX
```

Full Res Report

The Full Res Report outputs a _full_res.bed file. This file is tab-delimited file and has as its first three columns the standard BED fields, and as its fourth column 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 Report

The cov_report report generates a _cov_report.bed file. This tab-delimited file has the standard BED fields as its first three columns. The column fields that follow are statistics calculated over the interval region specified on the same record line. The additional columns are as follows:

- total_cvg—The total coverage value.
- mean_cvg—The mean coverage value.
- Q1_cvg—The lower quartile (25th percentile) coverage value.
- 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.

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
…
```

MAPQ and BQ Coverage Filtering

Coverage filtering is implemented to allow filtering out of reads below a certain mapping quality, and/or not counting bases below a certain base call quality in any read, over a specified coverage region. This filtering is applied to all reports for this region.

The coverage filtering can be enabled by using one of the *--qc-coverage-filters-i* options, where i can equal 1, 2, or 3, in combination with the associated *--qc-coverage-region-i* option:

- *--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.
- When metrics compression is enabled, a _full_res.bw file is output instead of the _full_res.bed.

BigWig compression of coverage metrics

The 1 bp resolution coverage metrics output bed file (_full_res.bed) can become very large. Bigwig format compression of this output can be enabled by setting the *--enable-metrics-compression* option to true.

Variant Quality Score Recalibration

The Variant Quality Score Recalibration (VQSR) module applies machine learning algorithms on an input VCF with training data from databases of known variants to produce a metric (VQSLOD). This metric is a better representation of whether a call is a true variant or a false variant. The VQSLOD metric is added to the INFO field of each variant call record and can be used later to filter out calls based on a threshold. The VQSLOD metric is the log odds ratio of the call being a true variant versus the call being a false variant.

Algorithm

The VQSR module attempts to build a Gaussian mixture model using the distribution of annotation values from a subset of high quality variant call sites. These variant call sites are determined from the specified resource files, or training sets, such as HapMap3 or Omni 2.5M SNP. From the model, each variant call site from the input set is then given a co-varying estimate based on the specified annotations, which indicates the likelihood that the call is a true genetic variant or not. The algorithm implements the variational inference version of the expectation maximization algorithm over a Gaussian mixture model. The model generation is done iteratively until convergence is met. Convergence may not be possible if there are too few sites in the input data set or the training sets.

Both a positive model and a negative model are generated for each pass. The negative model is generated from the worst performing sites via the *--vqsr-lod-cutoff* threshold. After both models are generated, the log likelihood that a call site was generated from each model is calculated, and then the log likelihood ratio is taken. The log likelihood ratio is annotated in the output VQSR VCF as VQSLOD under the INFO column.

You can further filter out call sites at a specific threshold by specifying tranche values, which indicate the target sensitivity levels. The VQSR module then calculates the minimum VQSLOD score required to meet this target sensitivity level. The *--vqsr-filter-level* option determines the threshold to filter out calls.

Usage and Settings

Enabling VQSR postprocessing adds only minutes to the DRAGEN pipeline for a whole genome analysis. The VQSLOD annotations are calculated for both SNPs and indels and are output in an additional VCF file, *<output-file-prefix>.vqsr.vcf*. To enable VQSR, use the *--enable-vqsr true* option on the DRAGEN command line.

The VQSR-specific options are as follows.

- *--enable-vqsr*

  Enables the VQSR post processing module. When used with *--enable-variant-caller* or *--enable-joint-genotyping*, the VQSR processing occurs after the variant calling stages. When used with *--vqsr-input*, the VQSR engine works in standalone mode to process VCF files.

- *--vqsr-config*

  Specifies the path to the VQSR configuration file. DRAGEN can optionally read in a configuration file that contains the VQSR options. VQSR options that are specified in the configuration file can be overridden with options specified on the command line. The configuration file is useful to store settings that are used across multiple runs, such as the tranche settings or resource files. The */opt/edico/config/dragen-VQSR.cfg* file provides an example of a VSQR configuration file.

- *--vqsr-input*

  Specifies the VQSR input VCF to be processed. If this option is specified on the DRAGEN command line, VQSR runs in standalone mode, which allows you to run VQSR on VCF files that were generated ahead of a time.

- *--vqsr-annotation*

  Specifies the annotations to be used for building the models as a comma-separated string that specifies the mode of operation followed by a list of annotations to be used under that mode. For example: <mode>,<annotation>,<annotation>…

  The <mode> is either *SNP* or *INDEL*. If only *SNP* is specified, then only SNPs are processed with VQSR. If only *INDEL* is specified, then only indels are processed. If both *SNP* and *INDEL* are specified, by using this option twice on the command line, then both SNPs and indels are processed.

  The list of annotations come from the INFO column of a VCF. You can specify up to eight annotations. These annotations and their associated values are used to build the model.

  The following is an example of VQSR annotations options:

  ```
  --vqsr-annotation SNP,DP,QD,FS,ReadPosRankSum,MQRankSum,MQ
  --vqsr-annotation INDEL,DP,QD,FS,ReadPosRankSum,MQRankSum
  ```

- *--vqsr-resource*

  Specifies the training resource files to be used for determining which variant call sites are true. This option can be specified multiple times to include multiple resource files, each with a different prior value. DRAGEN does not distinguish between truth or training resource files. All resource files will be used for both truth and training.

The format of this option is a comma-separated string that specifies the mode of operation, the prior value to weight this resource, then the path the resource file. For example: <mode>,<prior>,<resource file>.

<mode> is either *SNP* or *INDEL*. The <mode> indicates how to apply the resource files.

<prior> applies a weight to the variant call sites from the specified resource file with a prior probability of being correct.

<resource file> provides the path and file name of the VCF file to be used as the resource file. For example:

```
--vqsr-resource "SNP,15.0,<path>/hapmap_3.3.vcf"
--vqsr-resource "SNP,12.0,<path>/1000G_omni2.5.vcf"
--vqsr-resource "SNP,10.0,<path>/1000G_phase1.snps.high_confidence.vcf"
--vqsr-resource "INDEL,12.0,<path>/Mills_and_1000G_gold_
   standard.indels.vcf"
```

- *--vqsr-tranche*

  Specifies the truth sensitivity levels to calculate the LOD cutoff values. These values are specified in percentages. This option can be specified multiple times with a different truth sensitivity level. If none are specified, the default values of 100.0, 99.99, 99.90, 99.0, and 90.0 are used. For example:

  ```
  --vqsr-tranche 100.0
  --vqsr-tranche 99.99
  --vqsr-tranche 99.90
  --vqsr-tranche 99.00
  --vqsr-tranche 90.00
  ```

- *--vqsr-filter-level*

  Specifies the truth sensitivity level as a percentage to filter out variant calls. From the truth sensitivity level, the corresponding minimum VQSLOD score is calculated, and all annotated calls that are below this threshold are marked as filtered. The FILTER field is marked as failing with the filter VQSLODThresholdSNP or VQSLODThresholdINDEL. If not specified, none of the calls are filtered. The filter value must be specified independently for SNPs and indels. For example:

  ```
  --vqsr-filter-level SNP,99.5
  --vqsr-filter-level INDEL,90.0
  ```

- *--vqsr-lod-cutoff*

  Specifies the LOD cutoff score for selecting which variant call sites to use in building the negative model. The default value is -5.0.

- *--vqsr-num-gaussians*

  Specifies the number of Gaussians to generate the positive and negative models. This option is specified as a comma-separated string of the following four integer values: <SNP positive>,<SNP negative>,<INDEL positive>,<INDEL negative>. If not specified, the default values of 8,2,4,2 are used.

  The number of Gaussians to use per model must be greater than 0 and at most 8. The number of positive Gaussians must be greater than the number of negative Gaussians. As an example, to generate the models with 6 Gaussians for SNP positive, 2 Gaussians for SNP negative, 4 Gaussians for indel positive, and 2 Gaussians for indel negative, use the following option:

```
--vqsr-num-gaussians 6,2,4,2
```

- *--output-directory*

  Specifies the output directory where output files are stored.

- *--output-file-prefix*

  Specifies the output file prefix for all output files.

VSQR Example Output

If you already have a VCF that needs to be annotated and filtered, the VQSR module can be run as a standalone tool. The following is an example of the output:

```
==================================================================
   DRAGEN Variant Quality Score Recalibration

   ==================================================================
   Input file: /home/username/input.vcf
   Output file: output/dragen.vqsr.vcf
   Tranches: 100, 99.99, 99.9, 99, 90
   Annotations:
     SNP: MQ FS QD MQRankSum ReadPosRankSum DP
     INDEL: FS QD MQRankSum ReadPosRankSum DP
   Priors and training files:
     Q15.0:/variant_dbases/hapmap_3.3.vcf
     Q12.0:/variant_dbases/Mills_and_1000G_gold_standard.indels.vcf
     Q12.0:/variant_dbases/1000G_omni2.5.vcf
     Q10.0:/variant_dbases/1000G_phase1.snps.high_confidence.vcf
   Number of Gaussians     SNP     INDEL
     Positive model:       8       4
     Negative model:       2       2

   ==================================================================
   Building SNP Training
   Set==================================================================

   Number of valid records in input file: 5266144
   Number of training records detected: 4170912
     WARNING: Annotation 'QD' was missing in 41 records.
     WARNING: Annotation 'ReadPosRankSum' was missing in 702218 records.
     WARNING: Annotation 'MQRankSum' was missing in 702185 records.
   Training set statistics:
     DP -  mean: 140.838   std dev: 24.8402
     MQ -  mean: 59.9042   std dev: 0.946757
     QD -  mean: 22.7226   std dev: 6.67918
     FS -  mean: 3.01561   std dev: 4.18792  ReadPosRankSum -      mean:
   0.73308   std dev: 0.964922

   MQRankSum -   mean: 0.220358  std dev: 0.893418

   Number of outliers removed from the full set: 363154 out of 5266144
   Number of outliers removed from training set: 12046 out of 4170912
```

```
==================================================================
Generating SNP Positive Model
==================================================================
Number of Gaussians: 8
Number of data points: 4158866

Initializing model with k-means algorithm
K-means stabilized after 86 iterations
Running expectation-maximization algorithm
.......................................................
Expectation-maximization algorithm converged after 112 iterations

Cluster weight assigned to Gaussian #0 is 0.0109316
Cluster weight assigned to Gaussian #1 is 0.00988
Cluster weight assigned to Gaussian #2 is 0.646337
Cluster weight assigned to Gaussian #3 is 0.160654Cluster weight
assigned to Gaussian #4 is 0.00700244

Cluster weight assigned to Gaussian #5 is 0.005172
Cluster weight assigned to Gaussian #6 is 0.00252354
Cluster weight assigned to Gaussian #7 is 0.157501

==================================================================
Building SNP Negative Training Set
==================================================================
LOD Cutoff: -5
Number of data points: 359971

==================================================================
Generating SNP Negative Model
==================================================================
Number of Gaussians: 2
Number of data points: 359971

Initializing model with k-means algorithm
K-means stabilized after 15 iterations
Running expectation-maximization algorithm
.............
Expectation-maximization algorithm converged after 25 iterations

Cluster weight assigned to Gaussian #0 is 0.366887
Cluster weight assigned to Gaussian #1 is 0.633116

==================================================================
Calculating SNP LOD Ratios
==================================================================
Minimum LOD: -1317.29
Maximum LOD: 21.9196
```

```
=================================================================
Calculating SNP Truth Sensitivity Tranches
=================================================================
Tranche ts = 100.00     minVQSLOD = -1317.2891
Tranche ts = 99.99      minVQSLOD = -2.1371
Tranche ts = 99.90      minVQSLOD = -1.1866
Tranche ts = 99.00      minVQSLOD = 0.0199
Tranche ts = 90.00      minVQSLOD = 15.9062

=================================================================
Building INDEL Training Set
=================================================================
Number of valid records in input file: 1156226
Number of training records detected: 440621
  WARNING: Annotation 'QD' was missing in 41 records.
  WARNING: Annotation 'ReadPosRankSum' was missing in 77316 records.
  WARNING: Annotation 'MQRankSum' was missing in 76319 records.
Training set statistics:
  DP -  mean: 137.2     std dev: 41.05
  QD -  mean: 18.83     std dev: 8.258
  FS -  mean: 2.88      std dev: 4.756
  ReadPosRankSum -      mean: 0.2631    std dev: 0.9896
  MQRankSum -   mean: 0.1943    std dev: 0.907


Number of outliers removed from the full set: 8150 out of 1156226
Number of outliers removed from training set: 686 out of 440621

=================================================================
Generating INDEL Positive Model
=================================================================
Number of Gaussians: 4
Number of data points: 439935

Initializing model with k-means algorithmK-means stabilized after 52
iterations

Running expectation-maximization algorithm
..................
Expectation-maximization algorithm converged after 36 iterations

Cluster weight assigned to Gaussian #0 is 0.02934
Cluster weight assigned to Gaussian #1 is 0.2792
Cluster weight assigned to Gaussian #2 is 0.3979
Cluster weight assigned to Gaussian #3 is 0.2936

=================================================================
Building INDEL Negative Training Set
=================================================================
LOD Cutoff: -5
```

```
Number of data points: 61977

==================================================================
Generating INDEL Negative Model
==================================================================
Number of Gaussians: 2
Number of data points: 61977

Initializing model with k-means algorithm
K-means stabilized after 13 iterations
Running expectation-maximization algorithm
...................
Expectation-maximization algorithm converged after 38 iterations

Cluster weight assigned to Gaussian #0 is 0.5351
Cluster weight assigned to Gaussian #1 is 0.4649

==================================================================
Calculating INDEL LOD Ratios
==================================================================
Minimum LOD: -679.9
Maximum LOD: 6.154

==================================================================
Calculating INDEL Truth Sensitivity Tranches
==================================================================
Tranche ts = 100.00     minVQSLOD = -679.9446
Tranche ts = 99.99      minVQSLOD = -3.7305
Tranche ts = 99.90      minVQSLOD = -1.4809
Tranche ts = 99.00      minVQSLOD = -0.1606
Tranche ts = 90.00      minVQSLOD = 1.6201

==================================================================
Merging SNP and INDEL Records
==================================================================
Number of records processed as SNPs: 5266144
Number of records processed as INDELs: 1156226

==================================================================
Generating Output VCF
==================================================================
Number of total records from input file: 6422370
Number of records annotated with VQSLOD: 6422370
VQSR annotated VCF written to: output/dragen.vqsr.vcf
```

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. DRAGEN cannot process a set of three or more homologous regions. Support for three or more homologous regions will be added in a future release.

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.

```
/opt/edico/bin/dragen \
-r $REF \
-1 $FQ1 \
-2 $FQ2 \
--output-dir $OUTPUT \
--output-file-prefix $PREFIX \
--enable-map-align=true \
--enable-sort=true \
--enable-duplicate-marking=true \
--vc-sample-name=test \
--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 (mapping position and/or mapping quality, and so on). 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.

- *--enable-vlrd*

  If set to true, VLRD is enabled for the DRAGEN pipeline.

- *--vc-target-bed*

  Specifies the input bed file. DRAGEN requires an input target bed file specifying the homologous regions that are to be processed by VLRD. This bed file has a special format that is required by VLRD to correctly process the homologous regions. The maximum region span processed by VLRD is 900 bp.

  For example:

  ```
  chr1    161497562   161498362   0   0
  chr1    161579204   161580004   0   0
  chr1    21750837     21751637   1   0
  chr1    21809355      21810155   1   1
  ```

  - 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.

  The DRAGEN installation package contains two VRLD 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.

- *--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.

Force Genotyping

DRAGEN now supports force genotyping (ForceGT) for Germline SNV 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 SNV variant calling in the V3 mode. The V1, V2, and V2+ modes are not supported.
- ForceGT is not supported for Somatic SNV variant calling.
- ForceGT variants do not propagate through Joint Genotyping.

ForceGT Input

DRAGEN software supports only a single ForceGT VCF input file, which must meet the following requirements

- Have the same reference contigs as the VCF used for variant calling.
- Be sorted by reference contig name and position.
- Be normalized (parsimonious and left-aligned).
- Contain no complex variants (variants that require more than one substitution / insertion / deletion to go from ref allele to alt allele). For example, any variant in the ForceGT VCF similar to the following results in undefined behavior in the DRAGEN software:

```
chrX 153592402 GTTGGGGATGCTGAC CACCCTGAAGGG
```

The following nonnormalized variants cause undefined behavior in the DRAGEN software:

- Not parsimonious: `chrX 153592402 GC GCG`
- Parsimonious representation: `chrX 153592403 C CG`

ForceGT Operation and Expected Outcome

When running Germline SNV variant calling with ForceGT, a single sample gVCF is generated, using the ForceGT VCF as input on the DRAGEN command line. The single sample gVCF output file contains all regular calls and the forceGT calls, 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 ForceGT call that was also called by the variant caller and filter field is PASS (common), the call is tagged with NML:FGT in the INFO field (NML denotes normal)
- 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 allows us to distinguish between 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 single sample gVCF file. The reported GT for the variants are as follows:

Condition	Reported GT
At a position with no coverage	./.
At a position with coverage but no reads supporting ALT allele	0/0
At a position with coverage and reads supporting ALT allele	0/0 or 0/1 or 1/1 or 1/2

At a position where the variant call made by default DRAGEN software is different from the one specified in the input ForceGT vcf, there are multiple entries at the same position in the output gVCF, as follows:

- One entry for the default DRAGEN variant call, and
- 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 gVCF file only contains ForceGT variants that overlap the bed file positions.

Unique Molecular Identifiers

DRAGEN version 3.3 includes a beta release of the UMI pipeline.

The DRAGEN UMI pipeline supports the TruSight Oncology (TSO) UMI Reagents which use unique molecular identifiers (UMIs) to lower the rate of PCR or sequencing errors aiding in the detection of rare and low frequency somatic variants present in DNA samples, such as cfDNA isolated from plasma. After libraries that contain UMIs are sequenced, the DRAGEN UMI pipeline aligns the reads, then collapses the sequences with shared UMIs (families) down to unique reads. This process enables the filtering of false positives, which are excluded from the collapsed reads, reducing the error rate. The resulting reads have higher per-base quality and lower noise from various sources.

To use the UMI pipeline, the input reads files must be from a paired-end run and must contain UMI tags in the read name field in the standard Illumina format. The following example shows the UMI for both reads of the pair appended to the end of the read name:

```
NDX550136:7:H2MTNBDXX:1:13302:3141:10799:AAGGATG+TCGGAGA
```

To enable read collapsing, use the *--enable-umi* and *--remove-duplicates* options. The following is an example DRAGEN command to run the UMI pipeline:

```
/opt/edico/bin/dragen \
   -r $REF \
   -1 $FQ1 \
   -2 $FQ2 \
   --output-dir $OUTPUT \
   --output-file-prefix $PREFIX \
   --enable-map-align=true \
   --enable-sort=true \
   --enable-umi=true \
   --remove-duplicates=true
```

Chapter 4 DRAGEN RNA Applications

The DRAGEN RNA Applications use the DRAGEN RNA-Seq spliced aligner. Most of the functionality and options described in *Host Software Options* on page 117, *DRAGEN Host Software* on page 6, and *DRAGEN DNA Applications* on page 15*DRAGEN DNA Applications* on page 15 are also applicable to RNA-Seq processing. At a minimum, it is advisable to first read through *Host Software Options* on page 117, and *DNA Mapping* on page 15 through *Duplicate Marking* on page 24 before proceeding.

RNA Mapping and Intron Handling

The following options control the mapping stage of the RNA spliced aligner.

- *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 changed parameter to option

  and very near annotated splice junctions are not likely to be detected. Higher values may increase mapping time and false detections.

RNA Aligning

The following options control the aligner stage of the RNA spliced aligner.

Smith-Waterman Alignment Scoring Options

Refer to *Smith-Waterman Alignment Scoring Settings* on page 17 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.

- *--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).

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 values for XS:A strand tag are as follows:

‘.’ (undefined), ‘+’ (forward strand), ‘-’ (reverse strand), or ‘*’ (ambiguous).

If the spliced alignment has an undefined strand or a conflicting strand, then the alignment can be suppressed 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.

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 17 and *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 about some possible splice junction. 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, expressing 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 int(-10*log10(1 - 1/NH), where the NH value is the number of hits (primary and secondary alignments) for that read.

Input Files

In addition to the standard input files (*.fastq, *.bam, and so on), DRAGEN can also take a gene annotations file as input. A gene annotations file aids in the detection of splice junctions during RNA-Seq mapping and aligning.

Gene Annotation File

To specify a gene annotation file, use the *-a* (*--annotation-file*) command line option. The input file must conform to the GTF/GFF specification (http://uswest.ensembl.org/info/website/upload/gff.html). 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
   "ENSG00000223972.4"; transcript_id "ENST00000456328.2"; …
   chr1    HAVANA  exon        11869   12227   .   +   .   gene_id
```

```
"ENSG00000223972.4"; transcript_id "ENST00000456328.2"; …
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=transcript:ENST00000456328; …
   1   havana          exon                    12613   12721       .   +
   .   Parent=transcript:ENST00000456328; …
   1   havana          exon                    13221   14409       .   +
   .   Parent=transcript:ENST00000456328; …
   …
```

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 are also written to an output file, *.sjdb.annotations.out.tab, which is viewable by the user. When forming the splice junctions from the input gene annotations file, a simple filter is used that discards any splice junctions 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 Mode

In addition to reading in GTF/GFF files for use as annotated splice junctions, the DRAGEN software is also capable of reading in an SJ.out.tab file (see *SJ.out.tab* on page 89). The SJ.out.tab file enables DRAGEN to run in a two-pass mode, where the splice junctions (denoted by the SJ.out.tab file) discovered in the first pass are used to guide the mapping and alignment of a second invocation of DRAGEN. This mode of operation is useful when a gene annotations file is not readily available for the data set.

Output Files

The output files generated when running DRAGEN in RNA mode are similar to those generated in DNA mode. RNA mode 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.

Alignments in BAM/SAM

The output BAM or SAM file meets the SAM specification and is compatible with downstream RNA-Seq analysis tools.

RNA-Seq SAM Tags

The following SAM tags are emitted alongside spliced alignments.

- **XS:A**–The XS tag denotes the strand orientation of an intron. See *Compatibility with Cufflinks* on page 87.
- **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.

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 intron motif: 0: noncanonical, 1: GT/AG, 2: CT/AC, 3: GC/AG, 4: CT/GC, 5: AT/AC, 6: GT/AT

5 0: unannotated, 1: annotated, only if an input gene annotations file was used

6 number of uniquely mapping reads spanning the splice junction

7 number of multimapping reads spanning the splice junction

8 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.

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
```

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. The detection parameters are highly configurable and allow for a trade-off in sensitivity and specificity. Furthermore, all potential candidates (unfiltered) are output, which you can use to perform your own filtering.

Running DRAGEN Gene Fusion

The DRAGEN Gene Fusion module can be run in parallel with a regular RNA-Seq map/align job. This additional module adds minimal processing to the overall run time, while providing additional information to your RNA-Seq experiments.

To enable the DRAGEN Gene Fusion module, set the *--enable-rna-gene-fusion* option to true in your current RNA-Seq command-line scripts. The DRAGEN Gene Fusion module requires the use of a gene annotations file in the format of a GTF or GFF file.

The following is an example command line for running an end to end RNA-Seq experiment.

```
/opt/edico/bin/dragen \
   " -r "$REF" \"
   " -1 "$FQ1" \"
   " -2 "$FQ2" \"
   " -a "$GTF" \"
   " --output-dir "$OUTPUT" \"
   " --output-file-prefix "$PREFIX" \"
    --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
    ===================================================================
    annotation-file:            ref_annot.gtf
    rna-gf-blast-pairs:         blast_pairs.outfmt6
    rna-gf-exon-snap:           50
    rna-gf-min-anchor:          25
    rna-gf-min-neighbor-dist:   15
    rna-gf-max-partners:        3
    rna-gf-min-score-ratio:     0.15
    rna-gf-min-support:         2
    rna-gf-min-support-be:      10
    rna-gf-restrict-genes       true

    ===================================================================
    Completed DRAGEN Gene Fusion Detection
    ===================================================================
    Chimeric alignments: 107923
    Total fusion candidates: 38 (2116 before filters)

    Time loading annotations:                            00:00:08.543
    Time running gene fusion:                            00:00:18.470
    Total runtime:                                       00:00:27.760
    ***********************************************************
    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 "$GTF" \"
   " --rna-gf-input-file "$INPUT_CHIMERIC" \"
   " --output-dir "$OUTPUT" \"
   " --output-file-prefix "$PREFIX" \"
   --enable-rna true \
   --enable-rna-gene-fusion true \
```

Gene Fusion Candidates

The detected gene fusion events are listed in the fusion_candidate output files. Two files reside in the output directory, *.fusion_candidates.preliminary and *.fusion_candidates.final. The preliminary file is a prefiltered list of candidate events detected. The final file contains the candidate events with a sufficiently high confidence after passing all filters (content of the file is described below). The files contain the following five columns:

- Fusion gene
- A score
- The two breakpoints
- The supporting reads

The breakpoints are with respect to the split-read breakpoints as observed from the alignments. The + or - in the breakpoint indicates which strand, with respect to the reference FASTA, that the gene is transcribed from. The read names are delimited by semicolons. The file is sorted by the score, which is an indicator of the number of reads supporting that fusion event.

```
#FusionGene         Score   LeftBreakpoint      RightBreakpoint
   ReadNames
BSG--AL021546.6     108     chr19:580461:+      chr12:120876182:+   R_1;R_
   2;R_3;   …
FIS1--PMEL          70      chr7:100884111:-    chr12:56351868:-    …
CBX3--G3BP2         57      chr7:26242642:+     chr4:76572341:-     …
ELOVL5--SF3B14      46      chr6:53139888:-     chr2:24291329:-
ATXN10--GORASP2     44      chr22:46134719:+    chr2:171804860:+
FADS3--MTOR         39      chr11:61646784:-    chr1:11184690:-
AKR1B1--HMGB1       33      chr7:134135538:-    chr13:31036849:-
```

Gene Fusion Options and Filters

Various filters are implemented to help mitigate the number of false positive gene fusion candidates. The following thresholds and options are configurable. These filters are applied and all candidates that still qualify are emitted in the **.fusion_candidates.final* output file. For all prefiltered fusion candidates, see the **.fusion_candidates.preliminary* output file.

- *--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. One method to generate this file is to follow the instructions as described in Fusion Filter.

- *--rna-gf-restrict-genes*

  When parsing the gene annotations file (GTF/GFF) for use in DRAGEN Gene Fusion, this option can be used to restrict the entries of interest to only protein-coding regions. Restricting the GTF to only the protein-coding genes reduces false positive rates in currently studied fusion events. The default value is true.
- *--rna-gf-min-support*

  Specifies the minimum number of supporting reads to call a gene fusion event. A breakpoint is considered a broken exon if it is not precisely at an intron/exon boundary. A double broken exon is when both breakpoints do not lie on an intron/exon boundary. The default value is 2.
- *--rna-gf-min-support-be*

  Specifies the minimum number of supporting reads to call a double broken exon gene fusion event. The default value is 10.
- *--rna-gf-min-anchor*

  Specifies the minimum anchor length to call a gene fusion event. The anchor length is the number of nucleotides adjacent to a breakpoint. This minimum value applies to both breakpoints for the fusion event. The default value is 25.
- *--rna-gf-min-neighbor-dist*

  Specifies the minimum distance to neighboring candidate fusion events for them to be considered unique. If two candidate breakpoints are within this distance, then they are merged and their relative scores and attributes combined. The default value is 15.
- *--rna-gf-min-score-ratio*

  Minimum score ratio for a fusion event, for the highest scoring event. For all isoforms of a gene fusion pair, only accept it as a separate fusion event if it scores above this threshold. The dominant isoform is used as the maximum score. The default value is 0.15 (must score at least 15% of the highest scoring isoform).
- *--rna-gf-max-partners*

  Specifies the maximum number of gene partners a single gene can fuse with. Spurious gene fusion events can be indicative of false positives if a single gene fuses with multiple partner genes. If the number of partners exceeds this value, then filter it out as a false positive. The default value is 3.
- *--rna-gf-exon-snap*

  Specifies the breakpoint distance to an exon boundary for snapping. Any split-reads with breakpoints near an exon boundary are automatically clustered together to support a fusion event at that exon boundary. If the split-read breakpoint distance to the boundary is greater than this threshold, then it is not pulled in.

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-seq dataset. First, it internally translates the genomic mapping of each read (read pair) to the corresponding transcript mappings. Then it 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 GC-bias and correct for it in the reported quantification results.

Running Quantification

To enable the quantification module, set the *--enable-rna-quantification* option to true in your current RNA-seq command-line scripts. In addition, quantification requires the use of a gene annotations file (GTF/GFF), which provides the genomic position of all transcripts to quantify. This is specified with the -a (or *--annotation-file*) option.

Currently, only paired-end RNA-seq libraries in standard (inward) orientation are supported.

Quantification Outputs

Transcript quantification results are reported in the <outputPrefix>.quant.sf file. This is a text file, in which each line lists results for one transcript. For example:

```
Name                Length  EffectiveLength     TPM             NumReads
ENST00000364415.1   116     12.3238             5.2328          1
ENST00000564138.1   2775    2105.58             1.28293         41.8885
```

- Name lists the transcriptID of the transcript
- Length is the length of the (spliced) transcript in basepairs
- EffectiveLength is the length as accessible to RNA-seq, accounting for insert-size and edge effects
- TPM is Transcripts per Million, which represents the expression of the transcript, normalized for transcript length and sequencing depth
- NumReads is the estimated number of reads from the transcript (not normalized).

This file can be used as input for differential gene expression using tools such as tximport and DESeq2.

Similarly, the <outputPrefix>.quant.genes.sf file contains quantification results at the gene level. These are produced by summing together all transcripts with the same geneID in the annotation (GTF). Length and EffectiveLength are the (expression-)weighted means of the individual transcripts in the gene.

Quantification Options

- *--enable-rna-quantification*

  If set to true, enables RNA quantification. Requires enable-rna to be true as well.
- *--rna-quantification-library-type*

  Specifies the type of RNA-seq library. Only paired-end libraries in standard (inward) orientation are currently supported.
  - IU—Unstranded library
  - ISR—Stranded library in which read2 matches the transcript strand (eg, TruSeq RNA)
  - ISF—Stranded library in which read1 matches the transcript strand
- *--rna-quantification-gc-bias*

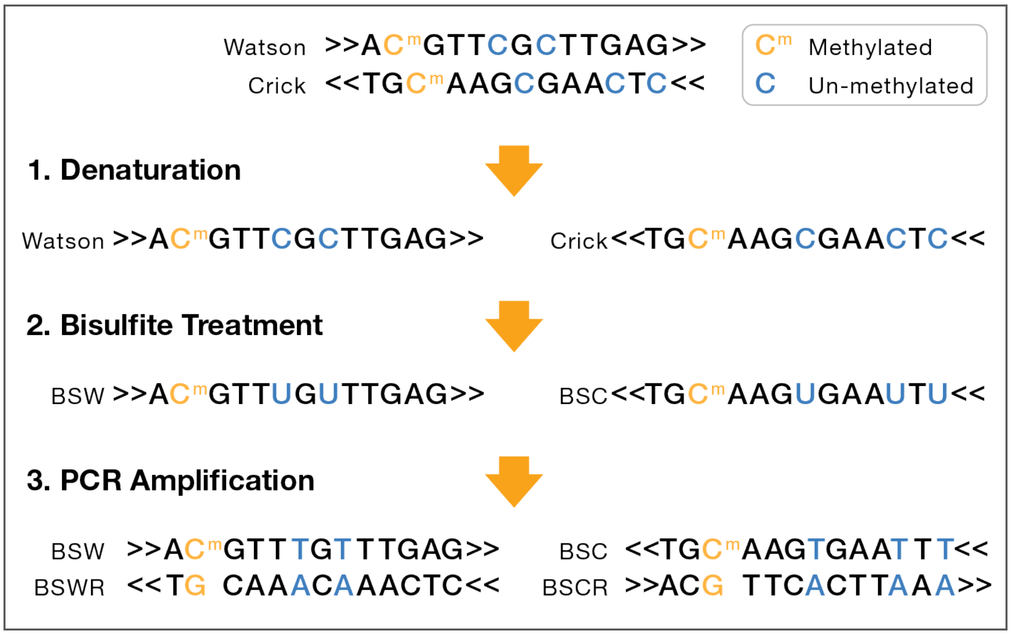
  If set to true, enables GC bias correction. This estimates the effect of transcript %GC on sequencing coverage and accounts for it when estimating expression.

Chapter 5 DRAGEN Methylation Applications

The epigenetic methylation of cytosine bases in DNA can have a dramatic effect on gene expression, and bisulfite sequencing is the gold standard 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:

Figure 10 Nondirectional Bisulfite Sequencing



- 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 (directional-complement protocol).

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

Standard DNA sequencing is used to produce sequencing reads. Reads containing more unconverted C's and G's complementary to unconverted C's are less drastically affected by the bisulfite treatment, and have a higher likelihood of mapping to the reference than reads with more bases affected. The standard protocol to minimize this mapping bias is to perform multiple alignments per read, where specific combinations of read and reference genome bases are converted *in-silico* prior to each alignment run. Each alignment run has a set of constraints and base-conversions that corresponds to one of the bisulfite+PCR strand types expected from the protocol. By comparing the alignment results across runs, you can determine the best alignment and most likely strand type for each read or read pair. This information is required for downstream methylation calling.

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, generating a single output BAM file with Bismark-compatible tags (XR, XG, and XM) that can be used in downstream pipelines, such as Bismark's methylation extraction scripts.

When the *--methylation-protocol* option is set to a valid value other than none, DRAGEN automatically produces the required set of alignments, each with its appropriate conversions on the reads, conversions on the reference, and constraints on whether reads must be forward-aligned or reverse complement (RC) aligned with the reference. 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.

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

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 that correspond to these strands (directional runs 1 & 2 above). However, 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 & 4 above).

The **directional-complement** protocol resembles the directional protocol in purpose, except that sequencing reads are only derived from the reverse-complement strands of the BSW and BSC strands. For the directional-complement protocol, very few good alignments are expected from runs 1 and 2, so DRAGEN is automatically tuned to a faster analysis mode for those runs.

For any protocol, you must run with a reference generated with *--ht-methylated* enabled, as described in *Pipeline Specific Hash Tables* on page 109.

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
```

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 corresponding to each base in the read's sequence. There is a period ('.') for each position not involving a cytosine, and a letter at cytosine positions. The letter indicates the context (CpG, CHG, CHH, or unknown), and the case indicates methylation, with upper-case positions being methylated and lower-case unmethylated. 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

To generate a genomewide cytosine methylation report, set *--methylation-generate-cytosine-report* to true. The position and strand of each C in the genome are given in the first three fields of the report. A record with a '-' in the strand field is for a G in the reference FASTA. The counts of methylated and unmethylated C's covering the position are given in the fourth and fifth fields, respectively. The C-context in the reference (CG, CHG, or CHH) is given in the sixth field, and the trinucleotide sequence context is given in the last field (eg, CCC, CGT, CGA, etc.). 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 a single-ended data with one table for each C-context, and six tables for a 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 C's (field 2) and unmethylated C's (field 3) that occur in the first read base position, restricting 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 skip reporting any C’s in the second read that overlaps the first read. In addition, 1-based coordinates are used for positions in both reports.

To match the bismark_methylation_extractor cytosine and M-bias reports generated by Bismark version 0.19.0, set the *--methylation-match-bismark* option to true. The ordering of records in Bismark and DRAGEN cytosine reports may differ. DRAGEN reports are sorted by genomic position.

Using Bismark for Methylation Calling

The recommended approach to methylation calling is to have DRAGEN automate the multiple required alignments, and add the XM, XR, XG tags as described previously. However, you can set *--enable-methylation-calling* to false to have DRAGEN generate a separate BAM file for each of the constraints and conversions by the methylation protocol. You can then feed these BAM files into a third-party tool for methylation calling. For example, you could modify Bismark to pipe reads from these BAM files instead of having Bismark internally run Bowtie or Bowtie2. Please contact Illumina Technical Support for assistance.

When you run in this mode, a single DRAGEN run produces multiple BAM files in the directory specified by *--output-directory*, each containing alignments in the same order as the input reads. It is not possible to 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. These 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 good alignments. For that reason, when running the nondirectional protocol, DRAGEN is automatically tuned to invest less work into the alignments on those runs.

Chapter 6 Preparing a Reference Genome

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

The DRAGEN system is shipped with reference genomes hg19 and GRCh37. Both reference genomes are preinstalled based on recommended settings for general applications. If you find that performance and mapping quality are adequate, there is a good chance that you can simply work with these 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 many overlapping seeds (subsequences or K-mers) from each read, and looks up those seeds in a hash table residing in memory on its PCIe card, to identify locations in the reference genome where the seeds match. Hash tables are ideal for extremely 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.

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, ie, 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, it is possible to populate less than all reference seeds 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*, and 50% population is specified 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

It is characteristic of hash tables that they are allocated a certain size, but always retain some empty records, so they are less than 100% occupied. A healthy amount of empty space is important for quick access to the DRAGEN hash table. Approximately 90% occupancy is a good upper bound. Empty space is important because records are pseudo-randomly placed in the hash table, resulting in an abnormally high number of records in some places. These congested regions can get quite large as the percentage of empty space approaches zero, 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 GB to 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, facilitating fast mapping without needing to check many alternative locations. But a longer seed is also more likely to overlap a deviation from the reference (variant or sequencing error), which prevents successful mapping by an exact match of that seed (although another seed from the read may 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 10 Seed Length Recommendations

Value for *-ht-seed-len*	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 match many locations in the reference genome. DRAGEN uses a unique mechanism called seed extension to successfully map such high-frequency seeds. When the software determines that a primary seed occurs at many reference locations, it extends the seed by some number of bases at both ends, to some greater length that is more unique in the reference.

For example, a 21-base primary seed may be extended by 7 bases at each end to a 35-base extended seed. A 21-base primary seed may match 100 places in the reference. But 35-base extensions of these 100 seed positions may divide into 40 groups of 1-3 identical 35-base seeds. Iterative seed extensions are also supported, and are automatically generated when a large set of identical primary seeds contains various subsets that are best resolved by different extension lengths.

The maximum extended seed length, by default equal to the primary seed length plus 128, can be controlled with the *--ht-max-ext-seed-len* option. For example, for short reads, it is advisable to set the maximum extended seed shorter than the read length, because extensions longer than the whole read can never match.

It is also possible to tune how aggressively seeds are extended using the following options (advanced usage):

- *--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. Faster mapping can be achieved using longer seed extensions to reduce seed hit frequencies, or more accurate mapping can be achieved by avoiding seed extensions or keeping extensions short, while tolerating the higher hit frequencies that result. 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 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, which is controlled with the *--ht-max-seed-freq* option. By default, the frequency limit is 16. In practice, when the software encounters a seed with higher frequency, it extends it to a sufficiently long secondary seed that the frequency of any particular extended seed pattern falls within the limit. However, if a maximum seed extension would still exceed the limit, the seed is rejected, and 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, for two reasons. First, because seeds are rejected only when extension fails, only extremely high-frequency primary seeds, typically with many thousands of matches are rejected. Such seeds are not very useful for mapping. Second, there are other seed positions to check 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 were rejected as high frequency, often this means that the entire read matches similarly well in many reference positions, so even if the read were mapped it would be an arbitrary choice, with very low or zero MAPQ.

Thus, the default frequency limit of 16 for *--ht-max-seed-freq* works well. However, it may be decreased or increased, up to a maximum of 256. A higher frequency limit tends to marginally increase the number of reads mapped (especially for short reads), but commonly the additional mapped reads 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, a *target* seed frequency can be specified with *--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 of 4 for *--ht-target-seed-freq* means that the software is biased toward generating shorter seed extensions than necessary to map seeds uniquely.

ALT-Aware Hash Tables

To enable ALT-aware mapping in DRAGEN, build GRch38 (and other references with ALT contigs) 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. The *--ht-alt-liftover* option specifies the path to the liftover file to build an ALT-aware hash table.

To override the liftover file requirement, set the *--ht-alt-aware-validate* option to false when building the hash tables and when running dragen.

The hash table builder detects when hg19 and hg38 references are missing decoy contigs and automatically adds them to the hash table. The decoys file is found at /opt/edico/liftover/hs_decoys.fa. Set the *--ht-suppress-decoys* option to true to suppress adding these decoys to the hash table.

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. It takes as input a FASTA file (multiple reference sequences being concatenated) and a preexisting output directory and generates the following set of files:

reference.bin	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.
hash_table.cmp	Compressed hash table. The hash table is decompressed and used by the DRAGEN mapper to look up primary seeds with length specified by the *--ht-seed-len* option and extended seeds of various lengths.
hash_table.cfg	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.
hash_table.cfg.bin	A binary version of hash_table.cfg used to configure the DRAGEN hardware.
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. It is not used by other tools.

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. Illumina 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.

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 this 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 GB to 64 GB, covering typical sizes for whole human genome, or k=26 for sizes from 4 GB to 16 GB.

The minimum primary seed length depends mainly on the reference genome size and complexity. It 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 may be smaller for shorter genomes, or higher for less complex (more repetitive) genomes. The uniqueness threshold of *--ht-seed-len 16* for the 3.1Gbp human genome can be understood intuitively because log4(3.1 G) ≈ 16, so it requires at least 16 choices from 4 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 they are less likely to overlap variants or sequencing errors, and because more of them fit in each read. So for mapping accuracy, shorter seeds are mainly better.

However, very short seeds can sometimes reduce mapping accuracy. Very short seeds often map to multiple reference positions, and 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 may be reported. Run time quality filters such as *--Aligner.alignn_min_score* can control the accuracy issues with very short seeds.

Speed Considerations

Shorter seeds tend to slow down mapping, because they map to more reference locations, resulting in more work such as Smith-Waterman alignments to determine the best result. This effect is most pronounced when primary seed length approaches the reference genome's uniqueness threshold, 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 (variants or sequencing errors) can chop the read into only short segments matching 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, just one SNP in the middle can block seeds longer than 18 bp from matching the reference. By contrast, in a 250 bp read, it takes 15 SNPs to exceed a 0.01% chance of blocking even 27 bp seeds.

- **Paired Ends**—The use of paired end reads can make longer seeds yield good mapping accuracy. 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 essentially 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 may 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 somewhere (even at low MAPQ), short seeds may be helpful. Some reads that do not match the reference well anywhere 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 (length specified by *--ht-seed-len*) 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*).

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.

When to Limit Seed Extension

The *--ht-max-ext-seed-len* option is recommended for short reads, eg, less than 50 bp. In such cases, it is helpful to limit seed extension to the read length minus a small margin, such as 1-4 bp. For example, with 36 bp reads, setting *--ht-max-ext-seed-len* to 35 might be appropriate. This ensures that the hash table builder does not plan a seed extension longer than the read causing seed extension and mapping to fail at run time, for seeds that could have fit within the read with shorter extensions.

While seed extension can be similarly limited for longer reads, eg, setting *--ht-max-ext-seed-len* to 99 for 100 bp reads, there is little utility in this 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 firm 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 this limit, it must be extended long enough 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 to 256. However, if this value is too low, hash table construction can fail because too many seed extensions are needed. The practical minimum for a whole human genome reference, other options being default, is 8.

Accuracy Considerations

Generally, a higher maximum hit frequency leads to more successful mapping. There are two reasons for this. First, a higher limit rejects fewer reference positions that cannot map under it. Second, a higher limit allows seed extensions to be shorter, improving the odds of exact seed matching without overlapping variants or sequencing errors.

However, as with very short seeds, allowing high hit counts can sometimes hurt mapping accuracy. Most of the seed hits in a large group are not to the true mapping location, and occasionally one of these noise hits may be reported due to imperfect scoring models. Also, the mapper limits the total number of reference positions it considers, and 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, resulting in more work, such as Smith-Waterman alignments, to determine the best result.

ALT-Aware Liftover File Options

The following options control See *ALT-Aware Hash Tables* on page 102 for more information on building a custom liftover file.

- *--ht-alt-liftover*

  The *--ht-alt-liftover* option specifies the path to the liftover file to build an ALT-aware hash table. This option is required when building from a reference with ALT contigs. SAM liftover files for hg38DH and hg19 are provided in /opt/edico/liftover.
- *--ht-alt-aware-validate*

  When building hash tables from a reference that contains ALT-contigs, building with a liftover file is required. To disable this requirement, set the *--ht-alt-aware-validate* option to false.
- *--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*

  Use the *--ht-suppress-decoys* option to suppress the use of the decoys file when building the hash table.

DRAGEN Software 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 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 GB < X ≤ 1 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. It makes sense to have these two options match, because building one hash table chunk requires one chunk space in memory and one thread to work on it. Nevertheless, 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 (option *--ht-mem-limit*). Using default table sizing is recommended and using *--ht-mem-limit* is the next best choice.

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, etc. Noninteger values are supported, eg, 2.5 yields 40% populated.

  Seeds from a whole human reference are easily 100% populated with 32 GB memory on DRAGEN boards. If a substantially larger reference genome is used, change this option.

- *--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. The idea is that 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 is also good for 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, it 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 are very useful in estimating MAPQ accurately for the alignments that are reported. They are never used outside of this context for reporting alignment positions, because that would result in biased coverage of locations that happened to be 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 pre-extension 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, 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, determining 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, 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. Current testing shows that zero (0) is appropriate for this parameter.

- *--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, and takes substantially more run time to execute the extensions. A higher value results in seed extension trees with fewer nodes, reaching from the root primary seed length to leaf extended seed lengths in fewer, larger steps.

Pipeline Specific Hash Tables

When building a hash table, DRAGEN configures the options to work for DNA-seq processing by default. To run RNA-Seq data, you must build an RNA-Seq hash table using the *--ht-build-rna-hashtable true* option. For an RNA-Seq alignment run, refer to the original *--output-directory*, not to the automatically generated subdirectory.

The CNV pipeline requires that the hash table be built with *--enable-cnv* set to true, which generates an additional kmer hashmap that is used in the CNV algorithm. Illumina recommends that that you always use the *--enable-cnv* option, in case you wish to perform CNV calling with the same hash table that is used for mapping and aligning.

DRAGEN methylation runs require building a special pair of hash tables with reference bases converted from C->T for one table, and G->A for the other. When running the hash table generation with the *--ht-methylated* option, these conversions are done automatically, and the converted hash tables are generated in a pair of subdirectories of the target directory specified with *--output-directory*. The subdirectories are named CT_converted and GA_converted, corresponding to the automatic base conversions. When using these hash tables for methylated alignment runs, refer to the original *--output-directory* and not to either of the automatically generated subdirectories.

These base conversions remove a significant amount of information from the hashtables, so you may find it necessary to tune the hash table parameters differently than you would in a conventional hash table build. The following options are recommended for building hash tables for mammalian species:

```
dragen --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
```

Chapter 7 Tools and Utilities

Illumina BCL Data Conversion

BCL format is the native output format of Illumina sequencing systems, and consists of a directory containing many data and metadata files. The data files are organized according to the sequencer’s flow cell layout, so converting this data to sample-separated FASTQ files is an intensive operation that can be time-consuming.

DRAGEN provides a custom implementation of the conversion software that is faster. To run this conversion, use the *--bcl-input-directory <BCL_ROOT>*, *--output-directory <DIR>*, and *--bcl-conversion-only true* options.

The DRAGENconversions implementation is designed to output FASTQ files that match Illumina's bcl2fastq2 2.19 output for HiSeq, MiSeq, HiSeq X, NextSeq, iSeq, and NovaSeq sequencers. DRAGEN supports most of the features of bcl2fastq2, including demultiplexing samples by barcode with optional mismatch tolerance, adapter sequence masking or trimming with adjustable matching stringency, and UMI sequence tagging and trimming. Sample sheet settings not supported by DRAGEN include FindAdapterWithIndels, CreateFastqForIndexReads, and ReverseComplement.

Command Line Options

The required options for BCL conversion in DRAGEN are shown in the following example command:

```
dragen --bcl-conversion-only --bcl-input-dir <…> --bcl-output-dir <…>
```

The following additional options can be specified on the command line:

- *--sample-sheet*–Specifies the path to SampleSheet.csv file. Optional if the SampleSheet.csv file is in the *--bcl-input-directory directory*
- *--strict-mode*–If set to true, *dragen* aborts if any files are missing. The default is false.
- *--first-tile-only*–If set to true, *dragen* only converts the first tile of input (for testing and debugging). The default is false.

The input BCL root directory and the output directory must be specified. The specified input path is not the BaseCalls directory but three levels higher, and should contain the following files and directories, among others:

```
Config\
Data\
Logs\
runParameters.xml
RunInfo.xml
```

The directory where output FASTQ files are to be stored is specified by the *--output-dir* option.

Sample Sheet Options

In addition to the command line options, which control the behavior of bcl conversion, the sample sheet accepts settings in the [Settings] section of the configuration file specifying how the samples are processed. The sample sheet settings for bcl conversion are as follows:

Option	Default	Value	Description
AdapterBehavior	trim	trim, mask	Whether adapter should be trimmed or masked.
AdapterRead1	None	Read 1 adapter sequence containing A, C, G, or T	The sequence to trim or mask from the end of read 1.
AdapterRead2	None	Read 2 adapter sequence containing A, C, G, or T	The sequence to trim or mask from the end of read 2.
AdapterStringency	0.9	Float between 0.5 and 1.0	The stringency for matching the read to the adapter using the sliding window algorithm.
BarcodeMismatchesIndex1	1	0, 1, or 2	The number of allowed mismatches between the first index read and index sequence.
BarcodeMismatchesIndex2	1	0, 1, or 2	The number of allowed mismatches between the second index read and index sequence.
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.
MaskShortReads	The minimum of 22 and MinimumTrimmedReadLength.	0 to MinimumTrimmedReadLength	Reads trimmed below this point become completely masked out.
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.

The OverrideCycles mask elements are semi-colon separated. For example:

```
OverrideCycles,N1Y150;I8;I7N1;Y151
```

Limitations

The BCL conversion in DRAGEN has been tested on, and is supported for, output from HiSeq 2000 and 2500, MiSeq, HiSeq X Ten, NextSeq, iSeq, and NovaSeq sequencer models with single-end and paired-end adapters. Related models may also operate correctly.

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
-m --swmaxtemp <n>	Maximum software alarm temperature (Celsius). Default is 85.
-i --swmintemp <n>	Minimum software alarm temperature (Celsius). Default is 75.
-H --hwmaxtemp <n>	Maximum hardware alarm temperature (Celsius). Default is 100.
-p --polltime <n>	Time between polling chip status register (seconds). Default is 60.
-l --logtime <n>	Log FPGA temp every n seconds. Default is 3600. Must be a multiple of polltime
-d --daemon	Detach and run as a daemon.
-h --help	Print help and exit.
-V --version	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

ID	Description	Monitor Action
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.

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_licd) 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.

Chapter 8 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 Identify if the System is Hanging* on page 116.
2. Collect diagnostic information after a hang, or a crash, as described in *Sending Diagnostic Information to Illumina Support* on page 116.
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 116.

How to Identify 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
```

This command takes several minutes to execute and reports the location where it has saved the diagnostic information in /tmp. Please include the report when you submit a support 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 116 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.

Appendix A Command Line Options

Host Software Options

The following options are in the default section of the configuration file. The default section does not have a section name (eg, [Aligner]) associated with it. The default section is at the top of the configuration file. Note that some mandatory fields must be specified on the command line and are not present in configuration files.

Name	Description	Command Line Equivalent	Range
alt-aware	Enables special processing for alt contigs, if alt liftover was used in hash table. Enabled by default if reference was built with liftover.	--alt-aware	true/false
annotation-file	Transcript annotation file (RNA).	--annotation-file, -a	
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.		true/false
bam-input	Aligned BAM file for input to the DRAGEN variant caller.	-b, --bam-input	
bcl-conversion-only	Perform Illumina BCL conversion to FASTQ format	--bcl-conversion-only	
bcl-input-directory	Input BCL directory for BCL conversion.	--bcl-input-directory	
sample-sheet	For BCL input, path to SampleSheet.csv file. Default location is the BCL root directory.	--sample-sheet	
build-hash-table	Generate a reference/hash table.	--build-hash-table	true/false
cram-input	CRAM file for input to the DRAGEN variant caller.	--cram-input	
dbsnp	Path to the variant annotation database VCF (or .vcf.gz) file.	--dbsnp	
enable-auto-multifile	Import subsequent segments of the *_001.{dbam,fastq} files.	--enable-auto-multifile	true/false
enable-bam-indexing	Enable generation of a BAI index file.	--enable-bam-indexing	true/false
enable-cnv	Enable copy number variant (CNV).	--enable-cnv	true/false
enable-duplicate-marking	Enable the flagging of duplicate output alignment records.	--enable-duplicate-marking	true/false
enable-map-align-output	Enables saving the output from the map/align stage. Default is true when only running map/align. Default is false if running the variant caller.	--enable-map-align-output	true/false
enable-methylation-calling	Whether to automatically add methylation-tags and output a single BAM for methylation protocols.		true/false
enable-rna	Enable processing of RNS-seq data.	--enable-rna	true/false
enable-sampling	Automatically detect paired-end parameters by running a sample through the mapper/aligner.		true/false

Name	Description	Command Line Equivalent	Range
enable-sort	Enable sorting after mapping/alignment.		true/false
enable-variant-caller	Enables the variant caller.	--enable-variant-caller	true/false
enable-vcf-compression	Enable compression of VCF output files. Default is true.		true/false
fastq-file1	FASTQ file to input to the DRAGEN pipeline (may be gzipped).	-1, --fasatq-file1	
fastq-file2	Second FASTQ file with paired-end reads for input.	-2, --fastq-file2	
fastq-list	CSV file that contains a list of FASTQ files to process.	--fastq-list	
fastq-list-all-samples	Enable/disable processing of all samples together, regardless of the RGSM value.	--fastq-list-all-samples	true/false
fastq-n-quality	Base call quality to output for N bases. Automatically added to fastq-n-quality for all output N's.	--fastq-n-quality	0 to 255
fastq-offset	FASTQ quality offset value.	--fastq-offset	33 or 64
filter-flags-from-output	Filter output alignments with any bits set in val present in the flags field. Hex and decimal values accepted.	--filter-flags-from-output	
first-tile-only	Only convert the first tile of each lane.	--first-tile-only	
force	Force overwrite of existing output file.	-f	
force-load-reference	Force loading of the reference and hash tables before starting the DRAGEN pipeline.	-l	
generate-md-tags	Whether to generate MD tags with alignment output records. Default is false.		true/false
generate-sa-tags	Whether to generate SA:Z tags for records that have chimeric/supplemental alignments.		true/false
generate-zs-tags	Whether to generate ZS tags for alignment output records. Default is false.		true/false
ht-alt-liftover	SAM format liftover file of alternate contigs in reference.	--ht-alt-liftover	
ht-alt-aware-validate	Disable requirement for a liftover file when building a hash table from a reference that contains alt-contigs.	--ht-alt-aware-validate	true/false
ht-build-rna-hashtable	Enable generation of RNA hash table. Default is false.	--ht-build-rna-hashtable	true/false
ht-cost-coeff-seed-freq	Cost coefficient of extended seed frequency.	--ht-cost-coeff-seed-freq	
ht-cost-coeff-seed-len	Cost coefficient of extended seed length.	--ht-cost-coeff-seed-len	
ht-cost-penalty-incr	Cost penalty to incrementally extend a seed another step.	--ht-cost-penalty-incr	
ht-cost-penalty	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	

Name	Description	Command Line Equivalent	Range
ht-max-dec-factor	Maximum decimation factor for seed thinning.	--ht-max-dec-factor	
ht-max-ext-incr	Maximum bases to extend a seed by in one step.	--ht-max-ext-incr	
ht-max-ext-seed-len	Maximum extended seed length.	-- ht-max-ext-seed-len	
ht-max-seed-freq	Maximum allowed frequency for a seed match after extension attempts.	--ht-max-seed-freq	1-256
ht-max-table-chunks	Maximum ~1 GB thread table chunks in memory at once.	--ht-max-table-chunks	
ht-mem-limit	Memory limit (hash table + reference), units B|KB|MB|GB.	--ht-mem-limit	
ht-methylated	Automatically generate C->T and G->A converted reference hashtables.	--ht-methylated	true/false
ht-num-threads	Maximum worker CPU threads for building hash table.	--ht-num-threads	
ht-rand-hit-extend	Include a random hit with each EXTEND record of this freq record.	--ht-rand-hit-extend	
ht-rand-hit-hifreq	Include a random hit with each HIFREQ record.	--ht-rand-hit-hifreq	
ht-ref-seed-interval	Number of positions per reference seed.	--ht-ref-seed-interval	
ht-reference	Reference file in .fasta format to be used to build a hash table.	--ht-reference	
ht-seed-len	Initial seed length to store in hash table.	--ht-seed-len	
ht-size	Size of hash table, units B|KB|MB|GB.	--ht-size	
ht-soft-seed-freq-cap	Soft seed frequency cap for thinning	--ht-soft-seed-freq-cap	
ht-suppress-decoys	Suppress the use of a decoys file when building a hash table.	--ht-suppress-decoys	
ht-target-seed-freq	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.		/ or . or :
interleaved	Interleaved paired-end reads in single FASTQ.	-i	
intermediate-results-dir	Directory to store intermediate results in (eg, sort partitions).		
lic-no-print	Suppress the license status message at the end of a run.	--lic-no-print	true/false
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	0/1
methylation-generate-cytosine-report	Generate a genomewide cytosine methylation report.	--methylation-generate-cytosine-report	true/false

Name	Description	Command Line Equivalent	Range
methylation-generate-mbias-report	Generate a per-sequencer-cycle methylation bias report.		true/false
methylation-match-bismark	If true, match bismark's tags exactly, including bugs.	--methylation-match-bismark	true/false
methylation-protocol	Library protocol for methylation analysis.		none / directional / nondirectional / directional-complement
num-threads	The number of processor threads to use.	-n, --num-threads	
output-directory	Output directory.	--output-directory	
output-file-prefix	Output file name prefix to use for all files generated by the pipeline.	--output-file-prefix	
output-format	The format of the output file from the map/align stage. Valid values are bam (the default), sam, or dbam (a proprietary binary format).	--output-format	BAM / SAM / DBAM
pair-by-name	Whether to shuffle the order of BAM input records such that paired-end mates are processed together.		
pair-suffix-delimiter	Change the delimiter character for suffixes.	--pair-suffix-delimiter	/ . :
preserve-bqsr-tags	Whether to preserve input BAM file's BI and BD flags. Note this may cause problems with hard clipping.		true/false
preserve-map-align-order	Produce output file that preserves original order of reads in the input file.		true/false
qc-coverage-region-1	First bed file to report coverage on.	--qc-coverage-region-1	
qc-coverage-region-2	Second bed file to report coverage on.	--qc-coverage-region-2	
qc-coverage-region-3	Third bed file to report coverage on.	--qc-coverage-region-3	
qc-coverage-reports-1	Types of reports requested for qc-coverage-region-1.	--qc-coverage-reports-1	full_res / cov_report
qc-coverage-reports-2	Types of reports requested for qc-coverage-region-2.	--qc-coverage-reports-2	full_res / cov_report
qc-coverage-reports-3	Types of reports requested for qc-coverage-region31.	--qc-coverage-reports-3	full_res / cov_report
ref-dir	Directory containing the reference hash table. This reference is automatically loaded to the DRAGEN card, if it is not already there.	-r, --ref-dir	
ref-sequence-filter	Output only reads mapping to this reference sequence.	--ref-sequence-filter	
remove-duplicates	If true, remove duplicate alignment records instead of just flagging them.		true/false
RGCN	Read group sequencing center name.	--RGCN	
RGCN-tumor	Read group sequencing center name for tumor input.	--RGCN-tumor	

Name	Description	Command Line Equivalent	Range
RGDS	Read group description.	--RGDS	
RGDS-tumor	Read group description for tumor input.	--RGDS-tumor	
RGDT	Read group run date.	--RGDT	
RGDT-tumor	Read group run date for tumor input.	--RGDT-tumor	
RGID	Read group ID.	--RGID	
RGID-tumor	Read group ID for tumor input.	--RGID-tumor	
RGLB	Read group library.	--RGLB	
RGLB-tumor	Read group library for tumor input.	--RGLB-tumor	
RGPI	Read group predicted insert size.	--RGPI	
RGPI-tumor	Read group predicted insert size for tumor input.	--RGPI-tumor	
RGPL	Read group sequencing technology.	--RGPL	
RGPL-tumor	Read group sequencing technology for tumor input.	--RGPL-tumor	
RGPU	Read group platform unit.	--RGPU	
RGPU-tumor	Read group platform unit for tumor input.	--RGPU-tumor	
RGSM	Read group sample name.	--RGSM	
RGSM-tumor	Read group sample name for tumor input.	--RGSM-tumor	
rna-ann-sj-min-len	Discard splice junctions that have length less than this value, during the generation of splice junctions from an annotations file (GTF/GFF/SJ.out.tab).		
rna-gf-input-file	An already generated .Chimeric.out.junction file. When this file is provided, DRAGEN Gene Fusion Module runs in standalone mode.	--rna-gf-input-file	
rna-mapq-unique	For compatibility with Cufflinks, enable this parameter with a nonzero value. Unique mappers have a MAPQ set to this value. Multimappers have a MAPQ of int(-10*log10(1 - 1 /NH)).		0, 1 to 255
sample-size	Number of reads to sample when enable-sampling is true.		
strict-mode	Abort if any files are missing	--strict-mode	
strip-input-qname-suffixes	Whether to strip read-index suffixes (eg, /1 and /2) from input QNAMEs.		true/false
tumor-bam-input	Aligned BAM file to for the DRAGEN variant caller in somatic mode.	--tumor-bam-input	
tumor-fastq-list	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	The sample ID for the list of FASTQ files specified by tumor-fastq-list.	--tumor-fastq-list-sample-id	

Name	Description	Command Line Equivalent	Range
tumor-fastq1	FASTQ file for the DRAGEN pipeline using the variant caller in somatic mode (may be gzipped).	--tumor-fastq1	
tumor-fastq2	Second FASTQ file with reads paired to tumor-fastq1 reads for the DRAGEN pipeline using the variant caller in somatic mode (may be gzipped).	--tumor-fastq2	
umi-enable	Enable UMI-based read processing.	--umi-enable	true/false
verbose	Enable verbose output from DRAGEN.	-v	
version	Print the version and exit.	-V	

Mapper Options

The following options are in the [Mapper] section of the configuration file. For more detailed information on these options, see *DNA Mapping* on page 15.

Name	Description	Command Line Equivalent	Range
ann-sj-max-indel	Maximum indel length to expect near an annotated splice junction.	--Mapper.ann-sj-max-indel	0 to 63
edit-chain-limit	For edit-mode 1 or 2: Maximum seed chain length in a read to qualify for seed editing.	--Mapper.edit-chain-limit	edit-chain-limit >= 0
edit-mode	0 = No edits, 1 = Chain len test, 2 = Paired chain len test, 3 = Edit all std seeds.	--Mapper.edit-mode	0 to 3
edit-read-len	For edit-mode 1 or 2: Read length in which to try edit-seed-num edited seeds.	--Mapper.edit-read-len	edit-read-len > 0
edit-seed-num	For edit-mode 1 or 2: Requested number of seeds per read to allow editing on.	--Mapper.edit-seed-num	edit-seed-num >= 0
enable-map-align	Enable use of BAM input files for mapper/aligner.	--enable-map-align	true/false
map-orientations	0=Normal, 1=No Rev Comp, 2=No Forward (paired end requires Normal).	--Mapper.map-orientations	0 to 2
max-intron-bases	Maximum intron length reported.	--Mapper.max-intron-bases	
min-intron-bases	Minimum reference deletion length reported as an intron.	--Mapper.min-intron-bases	
seed-density	Requested density of seeds from reads queried in the hash table	--Mapper.seed-density	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 17

Name	Description	Command Line Equivalent	Value
aln-min-score	(signed) Minimum alignment score to report; baseline for MAPQ. When using local alignments (global = 0), aln-min-score is computed by the host software as "22 * match-score". When using global alignments (global = 1), aln-min-score is set to -1000000. Host software computation may be overridden by setting aln-min-score in configuration file.	--Aligner.aln-min-score	−2,147,483,648 to 2,147,483,647
dedup-min-qual	Minimum base quality for calculating read quality metric for deduplication.	--Aligner.dedup-min-qual	0-63
en-alt-hap-aln	Allows chimeric 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	Score penalty for gap extension.	--Aligner.gap-ext-pen	0-15
gap-open-pen	Score penalty for opening a gap (insertion or deletion).	gap-open-pen	0-127
global	If alignment is global (Needleman-Wunsch) rather than local (Smith-Waterman).	--Aligner.global	0-1
hard-clips	Flags for hard clipping: [0] primary, [1] supplementary, [2] secondary.	--Aligner.hard-clips	3 bits
map-orientations	Constrain orientations to accept forward-only, reverse-complement only, or any alignments.	--Aligner.map-orientations	0 (any) 1 (forward only) 2 (reverse only)
mapq-max	Ceiling on reported MAPQ.	--Aligner.mapq-max	0 to 255
match-n-score	(signed) Score increment for matching a reference 'N' nucleotide IUB code.	--Aligner.match-n-score	-16-15
match-score	Score increment for matching reference nucleotide.	--Aligner.match-score	When global = 0, match-score > 0; When global = 1, match-score >= 0
min-score-coeff	Adjustment to aln-min-score per read base.	--Aligner. min-score-coeff	-64-63.999
mismatch-pen	Score penalty for a mismatch.	--Aligner.mismatch-pen	0-63
no-unpaired	If only properly paired alignments should be reported for paired reads.	--Aligner. no-unpaired	0-1
pe-max-penalty	Maximum pairing score penalty, for unpaired or distant ends.	--Aligner.pe-max-penalty	0-255
pe-orientation	Expected paired-end orientation: 0=FR, 1=RF, 2=FF.	--Aligner.pe-orientation	0, 1, 2

Name	Description	Command Line Equivalent	Value
sec-aligns	Maximum secondary (suboptimal) alignments to report per read.	--Aligner.sec-aligns	0-30
sec-aligns-hard	Set to force unmapped when not all secondary alignments can be output.	--Aligner.sec-aligns-hard	0-1
sec-phred-delta	Only secondary alignments with likelihood within this Phred of the primary are reported.	--Aligner.sec-phred-delta	0-255
sec-score-delta	Secondary aligns allowed with pair score no more than this far below primary.	--Aligner. sec-score-delta	
supp-aligns	Maximum supplementary (chimeric) alignments to report per read.	--Aligner.supp-aligns	0-30
supp-as-sec	If supplementary alignments should be reported with secondary flag.	--Aligner.supp-as-sec	0-1
supp-min-score-adj	Amount to increase minimum alignment score for supplementary alignments. This score is computed by host software as "8 * match-score" for DNA, and is default 0 for RNA.	--Aligner. supp-min-score-adj	
unclip-score	Score bonus for reaching each edge of the read.	--Aligner.unclip-score	0-127
unpaired-pen	Penalty for unpaired alignments in Phred scale.	--Aligner.unpaired-pen	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* on page 21. These options are part of the [Aligner] section of the configuration file.

Option	Description	Command Line Equivalent	Value
pe-stat-mean-insert	Average template length.		0-65535
pe-stat-mean-read-len	Average read length.		0-65535
pe-stat-quartiles-insert	A comma-delimited trio of numbers specifying 25th, 50th, and 75th percentile template lengths.		0-65535
pe-stat-stddev-insert	Standard deviation of template length distribution.		0-65535

Variant Caller Options

The following options are in the Variant Caller section of the configuration file. For more information on these options, see *Variant Caller Options* on page 27.

Name	Description	Command Line Equivalent	Value
cosmic	A COSMIC VCF input file.	--cosmic	
cnv-cbs-alpha	Significance level for the test to accept change points. Default is 0.01.	cnv-cbs-alpha	

Name	Description	Command Line Equivalent	Value
cnv-cbs-eta	Type I error rate of the sequential boundary for early stopping when using the permutation method. Default is 0.05.	--cnv-cbs-eta	
cnv-cbs-kmax	Maximum width of smaller segment for permutation. Default is 25.	--cnv-cbs-kmax	
cnv-cbs-min-width	Minimum number of markers for a changed segment. Default is 2.	--cnv-cbs-min-width	
cnv-cbs-nmin	Minimum length of data for maximum statistic approximation. Default is 200.	--cnv-cbs-nmin	
cnv-cbs-nperm	Number of permutations used for p-value computation. Default is 10000.	--cnv-cbs-nperm	
cnv-cbs-trim	Proportion of data to be trimmed for variance calculations. Default is 0.025.	--cnv-cbs-trim	
cnv-counts-method	Counting method for an alignment to be counted in a target bin. Default is overlap for panel of normal approach. For self normalization, the default is start.	--cnv-counts-method	midpoint / start / overlap
cnv-enable-gcbias-correction	Enable/disable GC bias correction.	--cnv-enable-gcbias-correction	true/false
cnv-enable-gcbias-smoothing	Enable/disable smoothing of the GC bias correction across adjacent GC bins with an exponential kernel. Default is true.	--cnv-enable-gcbias-smoothing	true/false
cnv-enable-ref-calls	When true, copy neutral (REF) calls are included in the output VCF.	--cnv-enable-ref-calls	true/false
cnv-enable-self-normalization	Enable/disable self-normalization.	--cnv-enable-self-normalization	true/false
cnv-enable-split-intervals	Splits up all target BED intervals into two equally spaced intervals. If this option is enabled, then all samples (case and panel of normals) must be executed with this option enabled. Default is false.	--cnv-enable-split-intervals	
cnv-enable-tracks	Enable/disable generation of track files that can be imported into IGV for viewing. Default is true.	--cnv-enable-tracks	true/false
cnv-extreme-percentile	Extreme median percentile value at which to filter out samples. Default is 2.5.	--cnv-extreme-percentile	
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. The default ratio is 0.2 (20% support).	--cnv-filter-bin-support-ratio	

Name	Description	Command Line Equivalent	Value
cnv-filter-copy-ratio	Minimum copy ratio threshold value centered about 1.0 at which a reported event is marked as PASS in the output VCF file. Default is 0.2	--cnv-filter-copy-ratio	
cnv-filter-de-novo-quality	Phred-scale threshold for calling an event as de novo in the proband.	--cnv-filter-de-novo-quality	
cnv-filter-length	Minimum event length in bases at which a reported event is marked as PASS in the output VCF file. Default is 10000.	--cnv-filter-length	
cnv-filter-qual	The QUAL value at which a reported event is marked as PASS in the output VCF file. Default is 10.	--cnv-filter-qual	
cnv-fpop-penalty	Penalty parameter for change point detection. Default is 0.03.	--cnv-fpop-penalty	
cnv-input	Sample input.	--cnv-input	
cnv-matched-normal	Target counts file of the matched normal sample.	--cnv-matched-normal	
cnv-max-percent-zero-samples	Threshold for filtering out targets with too many zero coverage samples. Default is 5%.	--cnv-max-percent-zero-samples	
cnv-max-percent-zero-targets	Threshold for filtering out samples with too many zero coverage targets. Default is 2.5%.	--cnv-max-percent-zero-targets	
cnv-merge-distance	Specifies the minimum number of base pairs between two segments that would allow them to be merged. The default value is 0, which means the segments must be directly adjacent.	--cnv-merge-distance	
cnv-merge-threshold	The maximum segment mean difference at which two adjacent segments should be merged. The segment mean is represented as a linear copy ratio value. Default is 0.2. Set to 0 to disable merging.	--cnv-merge-threshold	
cnv-min-mapq	Enable or disable splitting all target BED intervals into two equally spaced intervals. When enabled, all samples (case and panel of normals) must be executed with this option enabled. Default is false.	--cnv-min-mapq	true/false
cnv-normals-file	A single file to be used in the panel of normals. Can be specified multiple times, once for each file.	--cnv-normals-file	
cnv-normals-list	A panel of normals file.	--cnv-normals-list	
cnv-num-gc-bins	Number of bins for GC bias correction. Each bin represents the GC content percentage. Default is 25.	--cnv-num-gc-bins	10 / 20 / 25/ 50 / 100

Name	Description	Command Line Equivalent	Value
cnv-ploidy	The normal ploidy value. Used for estimation of the copy number value emitted in the output VCF file. Default is 2.	--cnv-ploidy	
cnv-qual-length-scale	Bias weighting factor to adjust QUAL estimates for segments with longer lengths. Advanced option that should not have to be modified. Default is 0.9303 (2-0.1).	--cnv-qual-length-scale	
cnv-qual-noise-scale	Bias weighting factor to adjust QUAL estimates based on sample variance. Advanced option that should not have to be modified. Default is 1.0.	--cnv-qual-noise-scale	
cnv-segmentation-mode	Segmentation algorithm to perform. The default value is slm or cb" depending on whether the intervals are whole genome intervals or targeted sequencing intervals.	--cnv-segmentation-mode	cbs / slm / hslm / fpop
cnv-slm-eta	Baseline probability that the mean process changes its value. Default is 1e-5.	--cnv-slm-eta	
cnv-slm-fw	Minimum number of data points for a CNV to be emitted. Default is 0.	--cnv-slm-fw	
cnv-slm-omega	Scaling parameter modulating relative weight between experimental/biological variance. Default is 0.3.	--cnv-slm-omega	
cnv-slm-stepeta	Distance normalization parameter. The default value is 10000. Only valid for “HSLM”.	--cnv-slm-stepeta	
cnv-target-bed	A properly formatted BED file that specifies the target intervals to sample coverage over. For use in WES analysis.	--cnv-target-bed	
cnv-target-factor-threshold	Percentage of median target factor threshold to filter out useable targets. The default value is 1% for whole genome processing and 10% for targeted sequencing processing.	--cnv-target-factor-threshold	
cnv-truncate-threshold	Extreme outliers are truncated based on this percent threshold. Default is 0.1%.	--cnv-truncate-threshold	
cnv-wgs-interval-width	Width of the sampling interval for CNV WGS processing. Default is 1000.	--cnv-wgs-interval-width	
cnv-wgs-skip-contig-list	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-wgs-skip-contig-list	

Name	Description	Command Line Equivalent	Value
enable-cnv-tracks	Enable/disable generation of bigwig and gff files.	--enable-cnv-tracks	true/false
enable-combinegvcfs	Enable/disable combine gVCF run.	--enable-combinegvcfs	true/false
enable-joint-genotyping	To enable combine gVCF run, set to true.	--enable-joint-genotyping	true/false
enable-multi-sample-gvcf	Enable/disable generation of a multisample gVCF file. If set to true, requires a combined gVCF file as input.	--enable-multi-sample-gvcf	true/false
enable-vqsr	Enable/disable the VQSR post processing module.	--enable-vqsr	true/false
panel-of-normals	The path to the panel of normals VCF file.	--panel-of-normals	
variant	The path to a single gVCF file. Multiple --*variant* options can be used on the command line, one for each gVCF. Up to 500 gVCFs are supported.	--variant	
pedigree-file	Specific to joint calling. The path to a PED pedigree file containing a structured description of the familial relationships between samples. Only pedigree files that contain trios are supported.	--pedigree-file	
variant-list	The path to a file containing a list of input gVCF files, one file per line, that need to be combined.	--variant-list	
vc-alt-allele-in-normal-filter	Set to false to disable the alt-allele-in-normal filter. Default is true.	--vc-alt-allele-in-normal-filter	true/false
vc-decoy-contigs	The path to a comma-separated list of contigs to skip during variant calling.	--vc-decoy-contigs	
vc-emit-ref-confidence	To enable base pair gVCF generation, set to BP_RESOLUTION. To enable banded gVCF generation, set to GVCF.	--vc-emit-ref-confidence	BP_RESOLUTION / GVCF
vc-enable-clustered-events-filter	Enable/disable the clustered events filter. The default is true.	--vc-enable-clustered-events-filter	true/false
vc-enable-decoy-contigs	Enable/disable variant calls on decoy contigs. Default is false.	--vc-enable-decoy-contigs	true/false
vc-enable-gatk-acceleration	Enable/disable running variant caller in GATK mode.	--vc-enable-gatk-acceleration	true/false
vc-enable-homologous-mapping-filter	Enable/disable the homologous mapping event filter. Default is true.	--vc-enable-homologous-mapping-filter	true/false
vc-enable-phasing	Enable variants to be phased when possible. Default is true.	--vc-enable-phasing	true/false
vc-enable-triallelic-filter	Enable the multi-allelic. Default is true.	-vc-enable-triallelic-filter	

Name	Description	Command Line Equivalent	Value
vc-gvcf-gq-bands	Define GQ bands for gVCF output. Default is 10, 20, 30, 40, 60, 80.	--vc-gvcf-gq-bands	
vc-hard-filter	Boolean expression for filtering variant calls. Default expression is: DRAGENHardQUAL:all: QUAL < 10.4139;LowDepth:all: DP < 1		Parameters in the expression can include QD, MQ, FS, MQRankSum, ReadPosRankSum, QUAL, DP, and GQ.
vc-limit-genomecov-output	Limit the size of the .genomecov.bed file that is generated. Default is false.	--vc-limit-genomecov-output	true/false
vc-max-alternate-alleles	Maximum number of ALT alleles that are output in a VCF or gVCF. Default is 6.	--vc-max-alternate-alleles	
vc-max-reads-per-active-region	Maximum number of reads per region for downsampling. Default is 1000.		
vc-max-reads-per-active-region	Maximum number of reads per region for downsampling; default is 1000.	--vc-max-reads-per-active-region	
vc-min-base-qual	Minimum base quality to be considered for variant calling. Default is 10.	--vc-min-base-qual	
vc-min-call-qual	Minimum variant call quality for emitting a call. Default is 30.	--vc-min-call-qual	
vc-min-read-qual	Minimum read quality (MAPQ) to be considered for variant calling. Default is 20.	--vc-min-read-qual	
vc-min-tumor- read-qual	Minimum tumor read quality (MAPQ) to be considered for variant calling.		
vc-min-reads-per-start-pos	Minimum number of reads per start position for downsampling. Default is 5.	--vc-min-reads-per-start-pos	
vc-remove-all-soft-clips	If set to true, variant caller does not use soft clips of reads to determine variants.	--vc-remove-all-soft-clips	true/false
vc-sample-name	Variant caller sample name, analogous field to RGSM.	--vc-sample-name	
vc-target-bed	Target regions BED file.	--vc-target-bed	
vc-target-coverage	Target coverage for downsampling. Default is 2000.	--vc-target-coverage	
vc-tlod-filter-threshold	Set the TLOD filter threshold. Default is 6.5.	--vc-tlod-filter-threshold	

Name	Description	Command Line Equivalent	Value
vqsr-annotation	A comma-separated string that specifies the annotations to use for building models. String format is *<mode>,<annotation>,<annotation>...* where mode is INDEL or SNP. See *Variant Quality Score Recalibration* on page 75.	--vqsr-annotation	
vqsr-config	The path to the VQSR configuration file.	--vqsr-config	
vqsr-filter-level	Specifies the truth sensitivity level as a percentage to filter out variant calls.	--vqsr-filter-level	
vqsr-input	Specifies the VQSR input VCF to be processed.	--vqsr-input	
vqsr-lod-cutoff	Specifies the LOD cutoff score for selecting which variant call sites to use in building the negative model. Default is -5.0.	--vqsr-lod-cutoff	
vqsr-num-gaussians	The number of Gaussians to generate the positive and negative models, specified as a comma-separated string of the following four integer values: <SNP positive>,<SNP negative>,<INDEL positive>,<INDEL negative>. If not specified, default values are 8, 2, 4, 2.	--vqsr-num-gaussians	
vqsr-resource	The training resource files to be used for determining which variant call sites are true, specified as a comma-separated string that specifies the mode of operation, the prior value to weight this resource, then the path the resource file. For example: <mode>,<prior>,<resource file>.	--vqsr-resource	
vqsr-tranche	Specifies the truth sensitivity levels to calculate the LOD cutoff values as percentages. This option can be specified multiple times with a different truth sensitivity level. If none are specified, the default values are 100.0, 99.99, 99.90, 99.0, and 90.0.	--vqsr-tranche	

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 Expansion Hunter* on page 60.

Parameter Name	Description	Command Line Equivalent	Range
enable	Enable or disable repeat expansion detection.	--repeat-genotype-enable	true/false
specs	The full path to the JSON file that contains the repeat variant catalog (specification) describing the loci to call.	--repeat-genotype-specs	

Technical Assistance

For technical assistance, contact Illumina Technical Support.

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

Illumina Customer Support Telephone Numbers

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

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

Product documentation—Available for download in PDF from the Illumina website. Go to support.illumina.com, select a product, then select **Documentation & Literature**.