Mar 2017

edico genome

DRAGEN™ User Guide

www.edicogenome.com

EDICO GENOME CORP. | 3344 North Torrey Pines Court, Plaza Level, La Jolla, CA 92037

Notice

Table of Contents

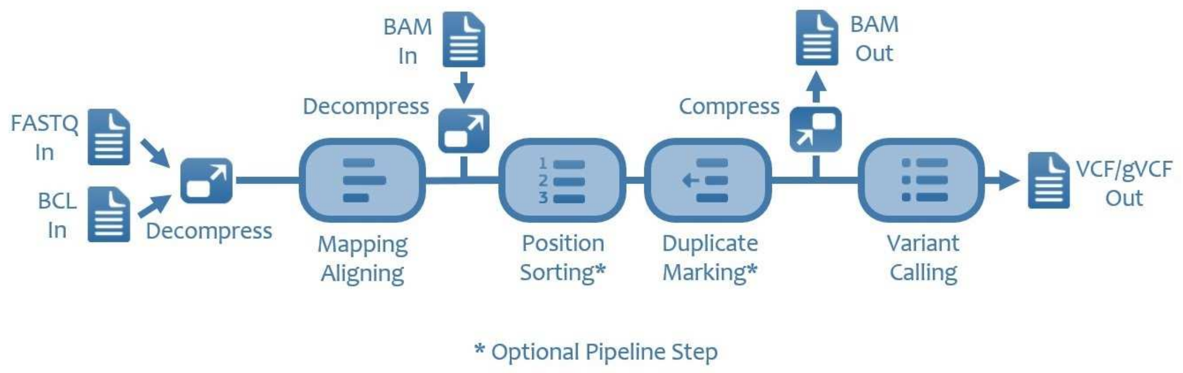
1 Introduction

1.1 DRAGEN^{TM} Platform

The DRAGEN^{TM} Bio-IT Platform is based on the highly reconfigurable DRAGEN^{TM} Bio-IT Processor, which is integrated on a PCIe card and is available in a pre-configured server that can be seamlessly integrated into a customer's NGS bioinformatics workflows. The platform can be loaded with highly optimized algorithms for many different NGS secondary analysis pipelines, such as whole genome or exome, RNAseq, methylome, microbiome and cancer. All user interaction is accomplished via DRAGEN^{TM} software that runs on the host server and manages all communication with the DRAGEN^{TM} board.

This user guide summarizes the technical aspects of the system, configuration parameters and detailed operating instructions for the software.

1.2 DRAGEN^{TM} GP - Genome Pipeline



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

This pipeline harnesses the tremendous power of the DRAGEN^{TM} Bio-It Platform and includes highly optimized algorithms for mapping, aligning, sorting, duplicate marking and haplotype variant calling. It also leverages platform features like compression and BCL conversion, together with the full set of platform tools.

Unlike all other secondary analysis methods, DRAGEN^{TM} GP does not reduce accuracy to achieve its speed improvements. In fact, accuracy for both SNPs and INDELs is improved over that of BWA-MEM+GATK-HC in side-by-side comparisons.

1.3 DRAGEN^{TM} TP - Transcriptome Pipeline

The DRAGEN^{TM} Transcriptome Pipeline shares many components with the Genome Pipeline, 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 Genome Pipeline, but detecting splice junctions (the joining of non-contiguous exons in RNA transcripts) near the mapped seeds and incorporating the junctions into alignments of the entire read sequences are specific to the Transcriptome Pipeline.

DRAGEN^{TM} uses hardware accelerated algorithms to map and align RNA-seq reads faster and more accurately than popular software tools. For instance, it can align 100 million, paired-end RNA-seq reads in about 3 minutes. With simulated, benchmark RNA-seq datasets, its splice junction sensitivity and specificity are unsurpassed.

The output BAM generated by the DRAGEN^{TM} Transcriptome Pipeline is compatible with popular downstream analysis tools.

1.4 DRAGEN^{TM} EP - Epigenome Pipeline

DRAGEN™ includes support for accelerating the series of map/align jobs typically used for methylation analysis of bisulfite sequencing data. This is described in Section 6.

1.5 System Updates

DRAGEN^{TM} has been architected to be a very flexible and extensible platform that is highly reconfigurable. This enables customers to take advantage of ongoing speed, performance, throughput and accuracy improvements. Customers will be able to remotely download updates to their DRAGEN^{TM} processors and software as part of their DRAGEN^{TM} subscription.

1.6 Additional Resources and Support

For additional information, resources, system updates and support, please visit the DRAGEN^{TM} Portal at http://dragen.edicogenome.com.

2 Getting Started Guide

In this section, we describe some initial tests you can run to check that your DRAGEN™ system is properly installed and configured. As a prerequisite, you will need to provide your DRAGEN™ server with adequate power and cooling, and connect it to a network that is fast enough to move your data on and off of the machine with adequate performance.

2.1 Running the System Check

After powering up, you can check that your DRAGEN™ system is in a good functional state by running ***/opt/edico/self_test/self_test.sh***. This shell script 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, and asserts that the alignments exactly match the expected results. Your machine ships with the test input FASTQ data for this script, located in */opt/edico/self_test*. The following example shows how to run the script and displays 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
---------------------------------

real    0m0.640s
user    0m0.047s
sys     0m0.604s
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 will read:

```
SELF TEST RESULT : FAIL
```

If you experience a FAIL result after running this test script immediately after powering up your DRAGEN™ system, please contact Edico Genome support via the DRAGEN™ Portal http://dragen.edicogenome.com.

2.2 Running Your Own Test

Once you are satisfied that your DRAGEN™ system is performing as expected, you are ready to run some of your own data through the machine.

2.2.1 Loading the Reference

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

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

You will need the location of the reference hash-table directory to be on the fast file IO drives.

On a newly shipped DRAGEN^{TM} server, a default reference hash table for hg19 is loaded at:

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

Therefore, on a newly shipped DRAGEN^{TM} server, the command to load reference genome hg19 would be:

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

This command will load the binary reference genome into memory on the DRAGEN^{TM} board, where it can be used for processing any number of input data sets; you will not need to reload it unless you restart the system or wish to switch to a different reference genome. This may take up to a minute to load.

DRAGEN^{TM} will check if the specified reference genome is already resident on the board. If it is, then DRAGEN^{TM} will automatically skip the upload of the reference genome. If for any reason you wish to force a reloading of the same reference genome, you can use the *force-load-reference (-l)* command line option.

Upon execution of the command to load the reference genome, the DRAGEN^{TM} software and hardware versions are printed to standard output. For example:

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

Upon successful completion of the loading of the reference genome, the following message will be printed to standard output:

```
DRAGEN finished normally
```

2.2.2 Identify Your Fastest File Systems

The DRAGEN^{TM} Bio-IT processor is so incredibly fast that you need to take care in planning where your input and output files reside. If your input or output files are located on a slow file system, then the overall performance of the system will be limited by the throughput of that single component.

Your system has been pre-configured with at least one fast file system consisting of a set of fast SSD disks grouped together with RAID-0 for performance. The 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*, since the failure of any of its constituent disks will lead to the loss of all data stored there. We therefore recommend that, before running, you copy your input data to */staging*, and direct your outputs there as well, but make sure to **copy all of your data in this staging area that you wish to preserve for archival purposes to another location**.

2.2.3 Process Your Input Data

Once you have loaded your reference genome and staged your input data, it is time to process your input FASTQ data. For example, you might process a single-ended FASTQ file as follows:

```
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
```

The full command-line options for this program are described in more detail below in Section 3: DRAGEN^{TM} Host Software.

3 DRAGEN^{TM} Host Software

Your main point of interaction with the DRAGEN^{TM} system is via the host software program *dragen*. This program is what you use for building and loading reference genomes onto the DRAGEN^{TM} platform and then decompressing, mapping, aligning, sorting, duplicate marking with optional removal and variant calling.

In the previous Section, we showed simple invocations of this program. In this Section we explore the full set of command-line options for the *dragen* software. Note that all of these options could also be set in the configuration file, as described below in Section 3.2. If an option is set in both places, the command-line values will override those in the configuration file.

3.1 Command-line Options

The usage summary for this program 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-reference <REF_FASTA> --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-reference <REF_FASTA> --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 Map/Align and Variant Caller from BCL** *(BCL to *.vcf)*

```
dragen -r <REF_DIR> --output-directory <OUT_DIR> --output-file-prefix <FILE_PREFIX> [options]
--bcl-input-dir <BCL_ROOT> --bcl-only-lane=<num> --enable-variant-caller=true --vc-reference
<REF_FASTA> --vc-sample-name <SAMPLE>
```

- **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
```

- **Options**

```
-1 [ --fastq-file1 ]              arg  FASTQ file to send to card (may be gzipped)
-2 [ --fastq-file2 ]              arg  Second FASTQ file with paired-end reads (may be gzipped)
-b [ --bam-input ]                arg  Input BAM file to send to card
                                       Directory with reference and hash tables
-r [ --ref-dir ]                  arg  Configuration file
-c [ --config-file ]              arg  Annotated splice junction file (RNA)
-a [ --ann-sj-file ]              arg  The number of processor threads to use
-n [ --num-threads ]              arg  Output filename prefix
--output-file-prefix              arg  Output directory
--output-directory                arg  Format of output file (SAM|BAM) (Default=BAM)
--output-format                   arg  Read group ID
--RGID                            arg  Read group library
--RGLB                            arg  Read group sequencing technology
--RGPL                            arg  Read group platform unit
--RGPU                            arg  Read group sample name
--RGSM                            arg  Read group sequencing center name
--RGCN                            arg  Read group description
--RGDS                            arg  Read group run date
--RGDT                            arg  Read group predicted insert size
--RGPI                            arg  Perform Illumina BCL conversion to FASTQ format
--bcl-conversion-only             arg  Input BCL directory for BCL conversion (must be specified
--bcl-input-dir                   arg  for BCL input)
                                       Output BCL directory for BCL conversion (must be
--bcl-output-dir                  arg  specified for BCL conversion to FASTQ)
                                       Barcode mismatch tolerance for BCL conversion (0-2,
--bcl-barcode-mismatches          arg  default 0)
                                       For BCL input, perform adapter sequence trimming using
--bcl-adapter-sequence            arg  specified file (HiSeq & MiSeq only)
                                       For BCL input, adapter sequence stringency: 0.5 to 1.0,
--bcl-adapter-stringency          arg  default 0.9
                                       For BCL input, include reads marked as 'failed' (filtered
--bcl-with-failed-reads           arg  out by default
                                       For BCL input, use specified mask for reads (Determined
--bcl-use-bases-mask              arg  from RunInfo.xml by default)
                                       For BCL input, path to SampleSheet.csv file (default
--bcl-sample-sheet                arg  searched for in BCL BaseCalls, then root directories)
                                       For BCL input, override autodetected sequencer type:
--bcl-seq-type                    arg  HiSeq, MiSeq, HiSeqXTen, & NextSeq supported
                                       For BCL input, disable EAMSS filtering (on by default for
--bcl-no-eamss                    arg  HiSeq & MiSeq, off and unsupported on HiSeqXTen & NextSeq
                                       For BCL input, reads trimmed below this # of bases become
--bcl-minimum-trimmed-read-length arg  masked at that point. Default 32, HiSeqXTen & NextSeq
                                       only

                                       For BCL input, completely mask out reads trimmed below
--bcl-mask-short-adapter-reads    arg  this # of bases. Default 10, HiSeqXTen & NextSeq only
                                       For BCL input, convert only specified lane number
--bcl-only-lane                   arg  (default all lanes, specify one lane for BCL streaming)
                                       For BCL input, convert & output only specified read
--bcl-only-read                   arg  number (default all non-index reads)
                                       BCL: concatenate fastq segments to one file per read
--bcl-consolidate-fastq-segments  arg  (default off)
                                       For BCL input, number of threads to use for conversion
--bcl-num-threads                 arg  (default 32)
                                       For BCL input, number of clusters to process per pass
--bcl-clusters-per-pass           arg  (default 131072)
                                       BCL: gzip-compress output fastq files (default off)
--bcl-compress-fastq              arg  BCL: use csv to indicate subset of samples to output
--bcl-sample-filter-list          arg  BCL: do not output unmatched reads to files (default off)
--bcl-only-matched-reads          arg  Import all fastq files in same directory with same sample
--combine-samples-by-name         arg  name as given file (even across lanes) (Default=false)
                                       Enable the output from mapper/aligner
--enable-map-align-output         arg  Import subsequent segments of *_001.fastq files
--enable-auto-multifile           arg  (Default=true)
                                       Output a .bai index file along with the output .bam
--enable-bam-compression          arg  Enable hardware-accelerated compression of BAM output
                                       files (Default=true)
                                       Enable sorting after mapping/alignment   (Default=true)
--enable-sort                     arg  Enable marking or removal of duplicate alignment records
--enable-duplicate-marking        arg  (Default=false)
```

```
                                          Remove duplicates instead of marking them with flag 0x400
--remove-duplicates               arg     (Default=false)
                                          FASTQ quality offset value. Set to 33 or 64 (Default=33)
--fastq-offset                    arg     Output only reads mapping to this reference sequence
--ref-sequence-filter             arg     Directory for sort intermediate results
--intermediate-results-dir        arg     Library protocol for methylation analysis.
--methylation-protocol            arg     (non|directional|non-directional|directional-complement)
                                          Preserve order of map/align input records
--preserve-map-align-order        arg     Enable the variant caller (Default=false)
--enable-variant-caller           arg     Reference file in FASTA format
--vc-reference                    arg     Target BED file
--vc-target-bed                   arg     Depth intervals BED file
--vc-depth-intervals-bed          arg     Enable depth of coverage calculations (Default=false)
--vc-enable-depth-of-coverage     arg     Variant caller sample name
--vc-sample-name                  arg     Target coverage for downsampling
--vc-target-coverage              arg     Minimum read quality (MAPQ)
--vc-min-read-qual                arg     Minimum base quality
--vc-min-base-qual                arg     Minimum variant call quality for emitting a call
--vc-min-call-qual                arg     Maximum number of reads per region for downsampling
--vc-max-reads-per-active-region
arg                                       Minimum number of reads per start position for
--vc-min-reads-per-start-pos      arg     downsampling
                                          Variant annotation database VCF (or .vcf.gz) file
--dbsnp                           arg     Generate a reference/hash table
--build-hash-table                arg     Reference in FASTA format
--ht-reference                    arg     Number of positions per reference seed
--ht-ref-seed-interval            arg     Initial seed length to store in hash table
--ht-seed-len                     arg     Maximum extended seed length
--ht-max-ext-seed-len             arg     Maximum allowed frequency for a seed match after
--ht-max-seed-freq                arg     extension attempts
                                          Target seed frequency for seed extension
--ht-target-seed-freq             arg     Soft seed frequency cap for thinning
--ht-soft-seed-freq-cap           arg     Maximum decimation factor for seed thinning
--ht-max-dec-factor               arg     Cost coefficient of extended seed length
--ht-cost-coeff-seed-len          arg     Cost coefficient of extended seed frequency
--ht-cost-coeff-seed-freq         arg     Cost penalty to extend a seed by any number of bases
--ht-cost-penalty                 arg     Cost penalty to incrementally extend a seed another step
--ht-cost-penalty-incr            arg     If set to true, generate C->T and G->A converted pair of
--ht-methylated                   arg     hashtables
                                          Include a random hit with each HIFREQ record
--ht-rand-hit-hifreq              arg     Include a random hit with each EXTEND record of this
--ht-rand-hit-extend              arg     frequency
                                          Worker threads for generating hash table
--ht-num-threads                  arg     Size of hash table, units B|KB|MB|GB
--ht-size arg                     arg     Memory limit (hash table + reference), units B|KB|MB|GB
--ht-mem-limit                    arg

                                          Force overwrite of existing output file
-f [ --force ]                            Load the reference, even if it appears to already be
-l [ --force-load-reference ]             loaded
                                          Interleaved paired-end reads in single FASTQ
-i [ --interleaved ]                      Print this help message
-h [ --help ]                             Be talkative
-v [ --verbose ]                          Print the version and exit
-V [ --version ]
```

Note that ‘ ‘ or ‘=’ may be used as a delimiter between a parameter name and its argument. In the following subsections, we describe these options in more detail, including any default values.

3.1.1 Reference Genome Options

Before you can use the DRAGEN^{TM} system for aligning reads, you must first load a reference genome and its associated hash tables onto the PCIe card. (Please see Section 4.6 for a description of how to preprocess a reference genome’s FASTA files into the native DRAGEN^{TM} binary reference and hash table formats.) You must also specify the directory containing the preprocessed binary reference and hash tables via the *–ref-dir (-r)* command-line option. This argument is *always* required. As previously noted, loading the reference genome and hash tables to DRAGEN™ card memory may be done separately from processing reads:

- **Load reference** *(-r [--ref-dir] arg)*

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

Use the force load reference command to force the reference genome to load even if it is already loaded:

- **Force load reference** *(-l [--force-load-reference] -r [--ref-dir] arg)*

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

The time needed to load the reference genome will depend on the size of the reference, but for typical recommended settings it will take approximately 30 seconds to one minute.

3.1.2 Operating Modes

DRAGEN™ has two primary modes of operation: mapper/aligner and variant caller. The DRAGEN™ system is capable of performing each mode independently or as an end-to-end solution. Additionally, the DRAGEN™ system allows the user to enable and disable decompression/sorting/duplicate marking/compression along the DRAGEN™ pipeline.

In the full pipeline mode of operation, DRAGEN™ will take as input unmapped reads (in *.fastq, *.bam, *.cram, or non-indexed BCL formats), perform decompression/mapping/aligning/sorting and optional duplicate marking and feed directly into the variant caller to produce a VCF file. In this mode, DRAGEN™ leverages parallel stages throughout the pipeline to drastically reduce the overall runtime. The full pipeline mode is executed when *--enable-variant-caller* is set to *true* and the input is in FASTQ, BAM, or BCL format.

In map/align mode, DRAGEN™ will take as input unmapped reads (in *.fastq, *.bam, *.cram, or non-indexed BCL format) and produce as output an aligned and sorted BAM or CRAM file. The reads can also be duplicate marked at the same time if *--enable-duplicate-marking* is set to *true* (it is *false* by default).

In variant caller mode, DRAGEN™ will take as input a mapped and aligned BAM file and produce a VCF file. This mode is controlled by the *--enable-variant-caller* option. If the BAM file is already sorted, that step can be skipped by setting the *--enable-sort* option to *false* (it is *true* by default). Note that the variant calling option is set to *false* by default. BAM files cannot be duplicate marked in the DRAGEN™ pipeline prior to variant calling if they have not already been marked. You must use the end-to-end mode of operation to take advantage of the mark-duplicates feature.

For RNA-Seq data, DRAGEN™ will utilize its RNA spliced aligner during the mapper/aligner stage. This distinction is transparent to the end user and the DRAGEN™ Bio-IT Processor will dynamically switch between the required mode of operation. To enable processing of RNA-seq data, set *--enable-rna* to *true*.

For processing of bisulfite methyl-seq data, the current version of DRAGEN™ does not currently make methylation calls. However, it does automate the processing of data for Lister and Cokus (aka directional and non-directional) protocols, producing a separate BAM output 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 using with *--ht-methylated* enabled, and run *dragen-* with the appropriate *-methylation-protocol* setting.

The remainder of this Section outlines the various options available to the user for finer control of the DRAGEN^{TM} pipeline.

3.1.3 Output Options

The output directory of generated files must be specified using the *--output-directory <out_dir>* command-line option. Additionally, you must specify the output file prefix using the *--output-file-prefix <out_prefix>* command-line option. DRAGEN^{TM} appends the appropriate file extension onto this prefix for each file that it generates. These two options, along with the reference hash table *-r [--ref-dir]* are mandatory. For brevity, the following examples will not explicitly list these 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 BAM format is chosen, BAM compression can be controlled with the '*--enable-bam-compression*' option. If the output file already exists, the software will issue a warning and exit unless the force overwrite *-f [--force]* option is used.

1. Output to **compressed BAM** file, **force overwrite** if file exists *(-f [--force])*

```
dragen ... -f
dragen ... -f --output-format=bam
dragen ... -f --output-format=bam --enable-bam-compression=true
```

2. Output to **uncompressed BAM** file, **force overwrite** if file exists *(-f [--force])*

```
dragen ... -f --output-format=bam --enable-bam-compression=false
```

3. Output to **SAM** file, **force overwrite** if file exists *(-f [--force])*

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

4. Output to **CRAM** file, **force overwrite** if file exists *(-f [--force])*

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

In addition, if you set the *--enable-bam-indexing* option to true, DRAGEN^{TM} will generate a BAI-format BAM index file. This will appear in the same output directory as the BAM file, but with a .bai extension.

DRAGEN™ is able to generate MD (“mismatch difference”) tags, as described in the BAM standard. However, because there is some minimal performance cost to generating these strings it is turned off by default. You can turn it on using the *--generate-md-tags* configuration variable.

3.1.4 Auto-generated MD5SUM for BAMs

An MD5SUM file will be generated automatically for BAM output files. This file will be in the same output directory and have the same name as the BAM output file, but with an .md5sum extension appended (example: whole_genome_run_123.bam.md5sum). This is a single-line text file which contains the md5sum of the BAM output file, and this md5sum exactly matches the output of the Linux 'md5sum' command.

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

3.1.5 Auto-generated METRICS

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

Mapping and aligning

Mapping and aligning metrics, similar to the metrics computed by the Samtools Flagstat command, are available on an aggregate level (over all input data), as well as on a per read group level. Methylation calling requires multiple mapping and aligning runs. In this case separate metrics are reported for each run.

Variant Calling

Metrics, similar to the metrics computed by RTG vcfstats, are reported on a per sample level. Metrics are reported for each sample in multi sample VCFs and gVCFs.

Duration

A breakdown of the run duration for each process (e.g. loading the reference, aligning reads, variant calling) is provided.

Output

The metrics are printed to the standard out in a human friendly format. The metric sections are identified by the headings:

```
MAPPING/ALIGNING METRICS
VARIANT CALLER METRICS
RUN TIME
```

In addition csv files are written to the run output directory. These files are written with the intent of being easy to parse and subsequently upload to a custom database for reference.

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

3.1.6 Input Options

The DRAGEN^{TM} system is capable of processing reads in FASTQ format or BAM/CRAM format. If your input FASTQ files end in ".gz", DRAGEN^{TM} will assume they were compressed with gzip and will automatically decompress them using hardware-accelerated decompression.

3.1.6.1 FASTQ input files

- **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
```

It is common for FASTQ samples to be segmented into multiple files to limit file size or to decrease the time to generate them. Both Illumina's bcl2fastq tool and the DRAGEN^{TM} BCL command use a common file naming convention similar to below (this differs somewhat for HiSeqX & NextSeq systems):

<SampleID>_<Index>_<Lane>_<Read>_<segment#>.fastq

For Example:

RDRS182520_CACACTGA_L001_R1_001.fastq
RDRS182520_CACACTGA_L001_R1_002.fastq
…
RDRS182520_CACACTGA_L001_R1_008.fastq

These files do not need to be concatenated in order to process together using DRAGEN^{TM}. In order to map/align any sample, provide it with the first file in the series (-1 <FileName>_001.fastq) and it will read all segment files in the sample consecutively. This works 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 may be turned off by including *--enable-auto-multifile=false* on the command line.

Beginning with version 1.3.3, DRAGEN^{TM} can also optionally read multiple files by the sample name given in the file name. This can be used to combine samples that may have been distributed across multiple BCL lanes or flow cells. To enable this feature, the '--combine-samples-by-name=true' option must be specified (the feature is off by default). If the FASTQ file(s) given on the command line use(s) the Casava 1.8 file naming convention shown above, and additional files in the same directory share the same sample name, those files and all of their segments will also be processed automatically. Note that sample name, read number, and file extension must match. Index barcode and lane number may differ.

Overall system performance will suffer if you do not make sure that your input files are located on a fast file system.

3.1.6.2 fastq-list input file

- **Specify sequence of fastq files in a csv file** *(--fastq-list option)*

```
dragen -r <ref_dir> --fastq-list <csv_file> --fastq-list-sample-id <Sample ID> --output-directory
<out_dir> --output-file-prefix <out_prefix>
```

The preferred way to specify multiple fastq input files is by a comma-separated-values (csv) file. This 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. The csv file is input via the '--fastq-list <filename>' command line option, and a csv file of the correct format is automatically generated during BCL conversion to FASTQ using DRAGEN. This 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.

The first line of the csv file indicates the title of each column, 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 whitespace or other extraneous characters.

Column titles are case sensitive, and the following titles must be present: RGID, RGSM, RGLB, Lane, Read1File, & Read2File. RGSM and RGLB refer to Sample ID & Library, respectively. Read1File and Read2File must be full path names to valid fastq input files. Every fastq file referenced in the csv list may be referenced only once. For input that is not paired-end, simply leave entries for 'Read2File' empty. All values in the 'Read2File' column must be non-empty and reference valid files, or they must all be empty.

When generating a BAM file using fastq-list input, one read group will be generated per unique 'RGID' value, and the BAM header will contain RG tags for ID (from RGID), SM (from RGSM), and LB (from RGLB). Additional tags can be indicated for each read group by adding a column whose title is four-characters, all-uppercase, and begins with 'RG'. For example, to add a PU (platform unit) tag, add a column named 'RGPU' and indicate the value for each read group in this column. All column titles must be unique.

Since a fastq-list file can contain more than one sample, typically the operator will want to specify which SampleID to process when using the fastq-list input method. If a fastq-list file contains only one unique RGSM entry, then no additional options need to be specified, and DRAGEN will process all files listed in the fastq-list file. If there are more than one unique RGSM entries in a fastq-list file, one of the following two options must also be specified in addition to --fastq-list <filename>.

The first option is to specify "--fastq-list-sample-id <SampleID>". This option filters the files to be processed during the run to those entries in the fastq-list file whose RGSM entry matches the SampleID value given on the command line, and should be used to process a specific sample among a larger group of samples. The second option, "--fastq-list-all-samples true", is to process all samples together in the same run, regardless of their RGSM value. 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.

Example fastq list csv file with required columns is given below:

```
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
```

3.1.6.3 BAM input files

BAM files may be used as input to the mapper/aligner by setting *--enable-map-align* to true. If you leave this setting set to false, you can still use the BAM file as input into the variant caller.

If you specify a BAM file as input, DRAGEN^{TM} will ignore any alignment information contained in the input file, and will output new alignments for all reads. If the input file contains paired-end reads, it is important to ensure that DRAGEN^{TM} knows to shuffle the input data such that pairs can be processed together. Other pipelines would require you in this situation to re-sort the input dataset by read name; DRAGEN vastly increases the speed of this operation by simply pairing the input reads, and sending them on to the mapper/aligner as soon as pairs are identified. You can enable this behavior using the *--pair-by-name* configuration variable. The default setting for this variable is true.

- **Single-ended input in one BAM file** *(-b option)*

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

- **Paired-end input in one BAM file** *(-b option)*

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

3.1.6.4 CRAM input

CRAM files may be used to input to mapper/aligner and variant caller. The functionality available with DRAGEN^{TM} using CRAM input, is the same as using BAM input. The following command line options are used for providing a CRAM input to either mapper/aligner or variant caller:

`cram-input`

The input cram file and path

`cram-reference`

The reference FASTA file to be used during CRAM compression and decompression. This should be the same reference that was used to build the hash table for mapping/aligning the reads. In addition, DRAGEN^TM requires that a FASTA .fai index file be located in the same folder as the CRAM FASTA reference.

- **Paired-end input in one CRAM file** *(--cram-input option)*

```
dragen -r <ref_dir> --cram-input <cram> --cram-reference <ref.fa> --output-directory <out_dir> --
output-file-prefix <out_prefix> --pair-by-name=true
```

3.1.6.5 BCL input

BCL is the output format of Illumina sequencers. Under limited circumstances, DRAGEN^TM can read directly from BCL for map-align operation, saving the time needed for conversion to FASTQ first. This can be done with the following limitations:

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

For other cases, DRAGEN^TM includes a very fast BCL to FASTQ converter (see section 9.1).

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

For additional options affecting BCL conversion, see section 9.1 below.

3.1.6.6 Handling of N bases

One of the techniques that DRAGEN^TM uses for optimizing the handling of sequences can lead to the overwriting of the base quality score assigned to "N" base calls. These qualities will all be overwritten with a fixed base quality, as specified by the *--fastq-n-quality* and *--fastq-offset* configuration settings. The default settings for these variables of 2 and 33 combine to match the Illumina minimum quality of 35 (ASCII character '#').

3.1.6.7 Read Names for Paired-End Reads

By a common convention, read names sometimes include suffixes (such as "/1" or "/2") indicating which end of a pair the read represents. For BAM input with the *--pair-by-name* option, DRAGEN™ must ignore these suffixes in order to find matching pair names. By default, DRAGEN™ uses the forward slash character as the delimiter for these suffixes, so it ignores the "/1" and "/2" when comparing names. You can control the delimiter character using the *--pair-suffix-delimiter* configuration variable; valid values include forward-slash (/), dot (.), and colon (:).

Note that that DRAGEN™ preserves the original read names -- including the suffixes -- in the output. If you wish for DRAGEN™ to add the suffixes, you can set the *–append-read-index-to-name* configuration variable to *true* to instruct DRAGEN™ to append suffixes, where the delimiter is determined by the *--pair-suffix-delimiter* variable. By default this is a slash, so /1 and /2 are added to the names.

3.1.6.8 Gene Annotation Files

When processing RNA-Seq data, a gene annotations file can be supplied to improve the accuracy of mapping and aligning stage (see section 5.3 for more details). The file is expected to conform to the GTF/GFF/GFF3 format specification and should list annotated transcripts that match the reference genome being mapped against. DRAGEN^{TM} can also take in the *SJ.out.tab* (described in section 5.4.2) as an annotations file to help guide the aligner in a “2-pass” mode of operation. To supply a gene annotations file, specify it with *-a* (*--ann-sj-file*) command line option.

3.1.6.9 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 you were to take such a read and pass it through a mapper/aligner with hard clipping enabled, the BI and/or BD tags can become invalid. Our recommendation is thus to strip these tags when using BAM files as input. You can accomplish this by setting the *--preserve-bqsr-tags* configuration option to *false*. If you wish to preserve the tags, DRAGEN™ will warn you to disable hard clipping.

3.1.7 Read Group Options

The current version of DRAGEN^{TM} assumes that all of the reads in a given FASTQ belong to the same read group. The DRAGEN^{TM} system is able to build a single @RG read group descriptor in the header of its output BAM, 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, e.g. 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^{TM} software will add an 'RG' tag to all of the output records to indicate that they are members of a read group. An example of a DRAGEN^{TM} command line including read group parameters is as follows:

```
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^{TM} system will allow FASTQ records to be individually tagged as belonging to different read groups.

3.1.8 License Options

- Removing the license status message at the end of the run (--lic-no-print option)

This option can be used to suppress the license status output at the end of a DRAGEN^{TM}:

```
DRAGEN Variant Caller completed in XXX sec
LICENSE_MSG| ***********************************************************
LICENSE_MSG| Licenses status:
LICENSE_MSG|   Genome      : used 779.0 Gbases since 2015-Jul-28 (779507910895 bases, unlimited)
LICENSE_MSG|
```

3.2 Configuration Files

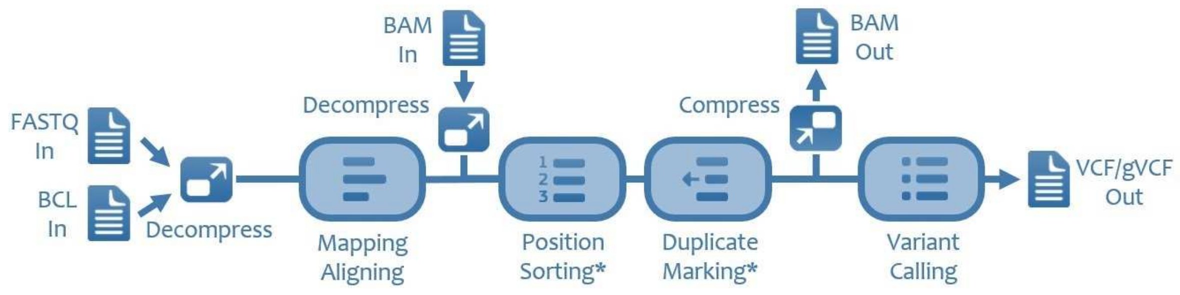
As mentioned earlier, user settings may be stored in a configuration file. The location of the default configuration file is ***/opt/edico/config/dragen-user-defaults.cfg***, but you may override this location by specifying the *--config-file (-c)* argument on the command line. The configuration file used for a given run supplies the default settings for that run, any of which may be overridden by that run's command-line arguments.

The recommended approach is to use the ***dragen-user-defaults.cfg*** file as a template by copying and modifying the copy for specific use case scenarios. It is best to enter rarely changed settings into the configuration file and to pass arguments that vary from run to run on the command line instead.

Additionally, there is a pre-made configuration file for analysis of noninvasive prenatal testing (NIPT) data. See Section 8.1 for more details.

In the following section, we describe some of these settings that are likely to stay the same for longer periods of time, and so are good candidates to be located in your configuration files.

4 DRAGEN^{TM} Genome Pipeline



* Optional Pipeline Step

4.1 Mapping

4.1.1 Seed Density Parameter

Seed Density

The *seed-density* parameter 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, i.e. (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 1 seed per 32 positions, or density 0.03125.

Accuracy Considerations

Generally, denser seed lookup patterns improve mapping accuracy. However, for modestly long reads (e.g. 50bp+) 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, once the seed mapping stage can run faster than the aligning stage, a sparser seed pattern will 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 parameter --*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. E.g. 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 won't systematically miss the seed positions populated from the reference. You don't have to worry, for example, about the mapper looking 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 = 0.5.

4.1.2 Map Orientations Parameter

The *map-orientations* parameter can restrict the orientation of read mapping to only forward in the reference genome, or only reverse-complemented. This parameter is for use in mapping reads for bisulfite methylation analysis. (Note that it will be set automatically based on the value set for --*methylation-protocol*.) The legal values are:

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.

4.1.3 Seed-Editing Parameters

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 (100bp+), because these 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 may, for example, be useful to increase mapping accuracy for short single-ended reads, with some cost in increased mapping time. The following parameters control seed editing.

`edit-mode, edit-chain-limit`

These parameters control when seed editing is used. Four edit-mode settings are available:

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, editing every seed that fails to match the reference exactly. 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, since there is already a promising mapping position.

Edit mode 1 simply 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, edit-read-len`

For edit modes 1 and 2, when the heuristic triggers seed editing, these parameters control how many seed positions are edited in the second pass over the read. Although exact seed mapping may 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 non-overlapping 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 parameters request, whenever seed editing is triggered, *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 may begin at offsets {0, 16, 32, 48, 64, 80} from the 5' end, consecutive seeds overlapping by 5 bases. Since 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 will be edited.

Note that seed editing is more expensive when the reference seed interval (build hash table parameter *-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.

4.2 Aligning

4.2.1 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 of 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, and a mismatch or indel imposes a penalty, on the score. 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^{TM} 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.

`global`

This option (0 or 1) controls whether alignment is forced to be end-to-end in the read. When 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 may be positive or negative. When 0, alignments may 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, etc.) can be clipped without severely decreasing the alignment score. In any event, setting *global=1* may not have its desired effect with longer reads, because insertions at or near the ends of a read can function as pseudo-clipping. 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 suspect of incorrect mapping if they cannot align well end-to-end.

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

`match-score`

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

`match-2-score`

This parameter 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 is a signed integer, from -16 to 15.

`match-3-score`

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

`match-n-score`

This parameter is the score for a read nucleotide matching an N code in the reference. This is a signed integer, from -16 to 15.

`mismatch-pen`

This parameter is the penalty (negative score) for a read nucleotide mismatching any reference nucleotide or IUPAC-IUB code (except 'N' which cannot mismatch). This 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`

This parameter is the penalty (negative score) for opening a gap, i.e. an insertion or deletion. This value is merely for a 0-base gap; it is always added to the gap length times gap-ext-pen. This 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`

This parameter is the penalty (negative score) for extending a gap, i.e. an insertion or deletion, by one base. This 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`

This parameter 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 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 leave unclip-score default 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 pseudo-clipping, as happens with global=1.

`no-unclip-score`

This option can be 0 or 1. When 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.

Observe that when unclip-score > 0 causes a Smith-Waterman local alignment to extend out to one or both ends of the read, the alignment score will stay the same or increase if *no-unclip-score = 0*, whereas it will stay the same or decrease 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.

Be careful when changing no-unclip-score to 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 *nounclipscore*.

aln-min-score

This parameter 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 is a signed integer (negative alignment scores are possible with *global = 0*).

Furthermore, *aln-min-score* 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 sub-optimal 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.

4.2.2 Paired-End Parameters

DRAGEN^{TM} is capable of processing 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.

pe-orientation

This parameter specifies the expected paired-end orientation. Only pairs with this orientation can be flagged as proper pairs. Legal values are:

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.

This parameter specifies how much alignment pair scores should be penalized when the two alignments are not in properly paired position or orientation. This also serves as the maximum pairing penalty for properly paired alignments with extreme insert lengths.

Note that unpaired-pen 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

This parameter 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 = 60*, 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; whereas pe-max-penalty affects only reported MAPQ for paired alignments.

4.2.3 Mean Insert Size Detection

When working with paired-end data, DRAGEN^{TM} must choose among the highest-quality alignments for the two ends to try to choose likely pairs. In order to do this, DRAGEN™ uses a Gaussian statistical model to evaluate the likelihood that a pair of alignments constitute a pair. This is based on the intuition that a particular library prep will tend to create fragments of roughly similar size, thus producing pairs whose insert lengths cluster well around some mean insert length.

If you know *a priori* 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 3 quartiles. (These are set using the following configuration parameters: *Aligner.pe-stat-mean-insert, Alinger.pe-stat-stddev-insert, Aligner.pe-stat-quartiles-insert,* and *Aligner.pe-stat-mean-read-len*.) Typically, however, it is preferably to allow DRAGEN™ to detect these characteristics automatically.

To enable automatic sampling of the insert-length distribution, set --*enable-sampling* to true. Then 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 that looks like this:

```
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 not
identical to TLEN)
```

When the number of sample pairs is very small, there is not enough information to characterize the distribution with high confidence. So in this situation, we apply 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, the host software outputs a message like this:

```
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 “standard deviation = 10000” is the default model. 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.

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

4.2.4 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, 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 is displayed:

```
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 may 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 = 0*.

4.2.5 Output Parameters

DRAGEN™ is able to track up to four independent alignments for each read. These alignments may be chimeric, meaning they map different regions of the read, or they may be sub-optimal mappings of the read to different areas of the alignment. Using common terminology, we refer to chimeric reads as “supplemental”, and to sub-optimal reads as “secondary.”

As DRAGEN™ tracks the four best alignments for a read, it gives priority to chimeric (supplemental) alignments over sub-optimal (secondary). In other words, it will track as many chimeric alignments as possible up to its limit of four; if there are fewer than four chimeric alignments, then the remaining slots will be used to track sub-optimal alignments. You can use the following configuration variables to control how many of each of these types of alignments you want DRAGEN™ to include in its output.

`mapq-max`

This parameter is 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 *mapq-max = 60*.

`supp-aligns, sec-aligns`

These parameters restrict the maximum number of supplementary (i.e. chimeric, SAM FLAG 0x800) alignments and secondary (i.e. suboptimal, SAM FLAG 0x100) alignments, respectively, that may be reported for each read.

A maximum of 31 alignments will be reported for any read total, including primary, supplementary, and secondary; therefore supp-aligns and sec-aligns can each range from 0 to 30. Supplementary alignments are tracked and output with higher priority than secondary ones.

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

`sec-phred-delta`

This parameter controls which secondary alignments are emitted based on its 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`

This parameter suppresses the output of all secondary alignments if there are more secondary alignments than may 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`

If this option is set to 1 instead of its default of 0, then supplementary (chimeric) alignments will be reported with SAM FLAG 0x100 instead of 0x800. This is for compatibility with some tools not supporting FLAG 0x800.

`hard-clips`

This parameter, ranging from 0 to 7, is considered as a field of 3 bits. Bit 0 is for primary alignments, bit 1 for supplementary alignments, and bit 2 for secondary alignments. Each bit determines whether local alignments of that type are reported with hard clipping (1) or soft clipping (0). The default is hard-clips = 6, meaning only primary alignments use soft clipping.

4.3 Sorting

By default the map/align system produces BAM sorted by reference sequence and position. This should relieve you of the requirement to run *samtools sort* or any equivalent post-processing command. The feature is on by default, but can be enabled/disabled using the command-line option:

```
--enable-sort=true
--enable-sort=false
```

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

4.4 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^{TM} system has the ability to mark or remove duplicate reads, producing a BAM file with duplicates marked in the FLAG field, or with duplicates entirely removed. In Edico Genome tests, enabling duplicate marking adds minimal runtime over and above that required to produce sorted BAM -- on the order of 1-2 minutes for a 30x whole human genome – a huge improvement over the long runtimes of open source tools.

4.4.1 The Duplicate Marking Algorithm

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

For two pairs to be duplicates they must have:

- 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.

Note that unmapped or secondary alignments are never marked as duplicates.

Once 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 just 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™ arbitrarily breaks the tie by choosing the one that came first in the input FASTQ as the winner.

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 variable with default value 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. Note that 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.

4.4.2 Duplicate Marking Limitations

There are several limitations to the DRAGEN™ duplicate marking implementation that you should be aware of:

- As noted above, when there are two duplicate reads or pairs with very close Phred sequence quality scores, DRAGEN™ may choose a different winner from that chosen by Picard. In our observations, 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, and library ID cannot at this time be used as a criterion for distinguishing non-duplicates.

4.4.3 Duplicate Marking Settings

The following configuration variables may be used to configure duplicate marking in DRAGEN™:

`enable-duplicate-marking`

If set to true, enables duplicate marking. Note that if this is turned on, the output will also be sorted, regardless of the *enable-sort* configuration setting.

`remove-duplicates`

If set to true, suppress the output of duplicate records. If set to false, set the 0x400 flag in the FLAG field of duplicate BAM records. Note that if this is turned on, then *enable-duplicate-marking* will be forced on.

`dedup-min-qual`

The Phred quality score below which a base should be excluded from the quality score calculation used for choosing among duplicate reads.

4.5 Variant Calling

The DRAGEN™ 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 not enabled by default. The user must specify variant calling with the *--enable-variant-caller=true* command line option.

4.5.1 The Variant Caller Algorithm

The DRAGEN™ Haplotype Caller operates in these major steps:

1. **Active Region Identification**: Areas where multiple reads disagree with the reference are identified, and windows around them ("active regions") are selected for processing.

2. **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 re-joining. 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, i.e. hypotheses for what the true DNA sequence may be on at least one strand.

3. **Haplotype Alignment**: Each extracted haplotype is Smith-Waterman aligned back to the reference genome, to determine what variation(s) from the reference it implies.

4. **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.

5. **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.

4.5.2 Variant Caller Settings

The following parameters are specific to the variant caller stage of the DRAGEN™ host software.

`enable-variant-caller`

If set to true, the variant caller stage will be enabled for the DRAGEN™ pipeline.

`vc-reference`

The reference FASTA file to be used during variant calling. This should be the same reference that was used to build the hash table for mapping/aligning the reads.

`vc-target-bed`

Restricts processing to regions specified in the BED file.

`vc-depth-intervals-bed`

Depth intervals BED file indicating which regions to collect depth of coverage statistics over.

`vc-enable-depth-of-coverage`

Enables the depth of coverage calculations over regions defined either in the target BED or depth intervals BED file.

`vc-sample-name`

The sample name being processed. This field is required when running the variant caller in stand-alone mode. In end-to-end mode you may use this option or use the --RGSM map/align option and that parameter will be passed through to the variant caller.

`vc-target-coverage`

The target coverage for downsampling. The default value is 2000.

`vc-min-read-qual`

The minimum read quality (MAPQ) to be considered for variant calling. The default value is 20.

`vc-min-base-qual`

The minimum base quality to be considered for variant calling. The default value is 10.

`vc-min-call-qual`

The minimum variant call quality for emitting a call. The default value is 30.

`vc-max-reads-per-active-region`

Maximum number of reads per region for downsampling. The default value is 1000.

vc-min-reads-per-start-pos

The minimum number of reads per start position for downsampling. The default value is 5.

vc-enable-gatk-acceleration

If set to true, the variant caller will run in GATK mode (concordant with GATK 3.6).

4.5.3 Somatic mode

Somatic mode allows detection of very low allele frequencies by using a different probabilistic model for determining whether a variant exists. This increase in sensitivity comes at the loss of specificity, so several filtering steps are applied after variant detection. The output is in the form of a vcf.

Filters:

clustered_events	Clustered events (>=2) were observed in the tumor, in a given active region, with distance >=3bp
homologous_mapping_event	3 or more events were observed in the tumor, in a given active region
t_lod_fstar	Variant does not meet likelihood threshold
triallelic site	Site filtered if there are two or more alt alleles at this location in the tumor
panel_of_normals	Seen in at least 2 samples in the panel of normals

4.5.3.1 Somatic mode

Somatic mode has different input command line options.

--tumor-fastq1/--tumor-fastq2

Is used to feed fastqs into the mapper aligner and somatic variant caller.

--tumor-bam-input

Is used to feed a mapped bam into the somatic variant caller.

--vc-enable-clustered-events-filter

Enables the clustered events filter. Default is true.

--vc-enable-homologous-mapping-filter

Enables the homologous mapping event filter. Default is true.

4.5.4 gVCF, CombineGVCF and JOINT VCF

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

gVCF, combine gVCF and joint calling is implemented in a multi-step 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 at all positions of the reference genome. Additional information is available that indicates how well the evidence (reads) supports the absence of variants (reference) or alternative alleles.

The combine gVCF step comes in next. This takes in multiple gVCF files as input and produces one combined gVCF file that represents all the input samples. The combine gVCF file can then be fed to joint caller.

During the second stage the multiple 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.

4.5.4.1 gVCF

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`

Needs to be set to "GVCF" to enable gVCF generation.

`--vc-gvcf-gq-bands`

Optionally used to define GQ bands for gVCF output. Example (and default): "5,20,60"

`--vc-max-alternate-alleles`

Maximum number of ALT alleles that will be output in a VCF or gVCF (Default=6)

4.5.4.2 Combine gVCF

Combine gVCF can be called once the gVCF files are created. The Combine gVCF is a tool used to merge input gVCF’s to generate a single gVCF file that represents all the input gVCF files. The following parameters are required by DRAGEN™ host software during combine gvcf.

`--enable-combinegvcfs`

Must be set to “true” to enable combine GVCF run.

`--output-directory`

Required output directory.

`--output-file-prefix`

Prefix used to label all output files.

`-r`

Hash table directory.

`--vc-reference`

Path to the reference fasta file.

`--variant`

Followed by the path to a single gVCF file. The command "--variant" needs to precede each additional gVCF that is added to the analysis. Up to 500 gVCFs are supported.

`--variant-list`

Followed by the path to a file that contains a list of input gVCF files that need to be combined, one variant file path per line.

4.5.4.3 JOINT CALLING

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

`--output-directory`

Required output directory.

`--output-file-prefix`

Prefix used to label all output files.

`--enable-joint-genotyping`

Must be set to true to enable joint genotyping.

`-r`

Hash table directory.

`--vc-reference`

Path to the reference fasta file.

`--variant`

Followed by the path to a single gVCF file. The command "--variant" needs to precede each additional gVCF that is added to the analysis.

`--variant-list`

Followed by the path to a file that contains a list of input gVCF files that need to be jointly called, one variant file path per line.

4.5.5 Variant Hard Filtering

DRAGEN™ supports basic filtering of variant calls as described in the VCF standard. You can apply any number of filters by providing the *--vc-hard-filter* configuration variable with a semicolon-delimited list of expressions in the following format:

<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 meanings of these expression elements are as follows:

- **filterID**: the name of the filter, which will be entered in the "FILTER" column of the VCF file for calls that are filtered by that expression.
- **snp/indel/all**: to which subset of variant calls the expression should be applied.
- **annotation ID**: the variant call record annotation whose 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 SNP's with FS < 2.1 or with MQ < 100, and will mark "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"

Note that in the current version, 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.

4.5.6 dbSNP Annotation

DRAGEN™ is able to look up its variant calls in a dbSNP database and add annotations for any matches that it finds there. You can enable the feature by setting the *--dbsnp* commandline 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, if the call matches a database entry for CHROM, POS, REF, and at least one ALT, then the ID for the matching database entry will be copied to the ID for that call in DRAGEN™'s 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 that used for alignment and variant calling. So please make sure that the contigs in your chosen annotation database match those in the alignment/variant calling reference.

4.5.7 Auto-generated MD5SUM for VCFs

An MD5SUM file will be generated automatically for VCF output files. This file will be in the same output directory and have the same name as the VCF output file, but with an .md5sum extension appended (example: whole_genome_run_123.vcf.md5sum). This is a single-line text file which contains the md5sum of the VCF output file, and this md5sum exactly matches the output of the Linux 'md5sum' command.

4.6 Depth of Coverage

The depth of coverage feature *--vc-enable-depth-of-coverage=true* collects coverage statistics across the reads from a BAM file. The feature outputs the coverage at every base position and mean, median, and quartile statistics over specified regions in an input depth-intervals BED file. This may, for example, provide useful analysis metrics for determining missed calls due to low coverage.

The coverage calculations are processed in parallel alongside variant calling and adds minimal processing time to the overall pipeline. The processing time is directly proportional to the interval regions of interest, which is specified via an input depth-intervals BED file. The depth of coverage feature cannot be executed when only map/aligning is performed. To use this feature, the user must enable variant calling with the -*-enable-variant-caller=true* command line option.

4.6.1 Usage

The feature is enabled with the command line option *--vc-enable-depth-of-coverage=true*. The regions over which the coverage statistics are collected must be specified via a target BED file *--vc-target-bed* or a depth intervals BED file *--vc-depth-intervals-bed*. The depth intervals BED file need not be the same as the target BED file. The BED files are expected to conform to the UCSC BED format. The first three columns should be *chrom*, *chromStart*, and *chromEnd*. The first base in a chromosome is 0, and the first 100 bases of a chromosome are defined as *chromStart=0*, *chromEnd=100*, spanning the bases 0-99.

The following is an example command line on top of the regular variant calling command line arguments that would enable the depth of coverage feature.

```
$ dragen ... -enable-variant-caller true \
    –vc-enable-depth-of-coverage true \
    –vc-depth-intervals-bed <BED>
```

4.6.1.1 BED Files

Due to the nature of processing the depth of coverage statistics in parallel with variant calling, further clarification on the different use cases and the BED files is explained in this sub section.

The target BED file (*--vc-target-bed*) is used to specify the regions over which variant calling should be performed. This is commonly used when the upstream chemistry is targeting specific genes or regions, such as in an exome capture experiment. The target BED file affects both the **.genomecov.bed* output file as well as the **.interval_summary* output file.

The depth intervals BED file (*--vc-depth-intervals-bed*) is used to specify the regions over which the depth of coverage statistics should be computed. The depth intervals BED file affects only the **.interval_summary* output file.

The various combinations are outlined below.

1. You would like to perform variant calling over the entire genome and would like to calculate depth of coverage statistics over specific regions. To do so, you would specify only a depth intervals BED file (*--vc-depth-intervals-bed)* indicating the intervals of interest for coverage statistics.
2. You would like to perform variant calling over a targeted region, and you would also like to calculate depth of coverage statistics over the **same** region. To do so, you need only specify a target BED file (*--vc-target-bed*). You may optionally provide the same BED file as a depth intervals BED file (*--vc-depth-intervals-bed*) but it is not necessary.
3. You would like to perform variant calling over a targeted region, and you would also like to calculate depth of coverage statistics over a **different** region. To do so, you would specify both a target BED file (*--vc-target-bed*) for variant calling as well as a depth intervals BED file (*--vc-depth-intervals-bed*) for the statistics calculations. In this scenario, only intervals which fall under both BED files will have statistics computed.

The following table further summarizes the combinations for *--vc-enable-depth-of-coverage=true*

	Input Files		**Output Files**	
Use Cases w/ Depth of Coverage	**Target BED**	**Depth Intervals BED**	***.genomecov.bed**	***.interval_summary**
Variant Calling is to be performed over the **entire genome** and the user wishes to collect depth of coverage metrics across a subset of regions. Those regions are specified via the vc-depth-intervals-bed file.	Not Set	Set	A depth map which indicates the coverage at each base for the alignments being processed. This file can be further limited to the **vc-depth-intervals-bed** regions if **--vc-limit-genomecov-output** is set to true.	Depth of coverage metrics for each interval specified in the **vc-depth-intervals** file.
Variant Calling is to be performed over a **targeted region** (such as an exome), and the user wishes to collect depth of coverage metrics over the same targeted region. The user need only specify a vc-target-bed file.	Set	Not Set	A depth map which indicates the coverage at each base for the alignments being processed, but only for those alignments contained within the **vc-target-bed** file.	Depth of coverage metrics for each interval specified in the **vc-target-bed** file.
Variant Calling is to be performed over a **targeted region** (such as an exome), but the user wishes to collect depth of coverage metrics over a different subset of regions. The user needs to specify a vc-target-bed file over which variant calling is performed, and a vc-depth-intervals-bed file over which the depth of coverage metrics are calculated.	Set	Set	A depth map which indicates the coverage at each base for the alignments being processed, but only for those alignments contained within the **vc-target-bed** file.	Depth of coverage metrics for each interval specified in the **vc-depth-intervals** file.
If vc-enable-depth-of-coverage is enabled, then at least one BED file must be specified otherwise the software cannot determine which intervals to calculate the metrics over.	Not Set	Not Set	Invalid	Invalid

4.6.2 Output Files

The depth of coverage output files are prefixed with the *--output-file-prefix* parameter, similar to all other output files from DRAGEN™. The files generated are *<prefix>.genomecov.bed* and *<prefix>.interval_summary*. The contents are described in the following sections.

4.6.2.1 *.genomecov.bed

The **.genomecov.bed* file is a tab delimited file which 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 which has a constant depth. If the depth changes then a new record is printed to the file. Alignments which have a mapping quality value of 0 are not counted towards the depth. To minimize the file size, if an interval has a depth of 0, then the record is not printed to the **.genomecov.bed* file.

The number of records emitted in the **.genomecov.bed* file are affected by whether or not a target BED file (*--vc-target-bed*) was specified. Only base positions which fall under the target BED regions will be present in the **.genomecov.bed* output file. If no target BED file was specified, then the *.genomecov.bed file is not filtered by any regions and will emit a record for every alignment position.

If the analysis is done on a whole genome, the *.genomecov.bed file may become quite large. To limit the file size, the user may specify --vc-limit-genomecov-output=true on the command line. Doing so will only emit records that overlap with the --vc-depth-intervals-bed regions of interest. This option is only applicable when running a whole genome analysis (no target BED specified). The default value of --vc-limit-genomecov-output is false.

The **.genomecov.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).

Example contents of the **.genomecov.bed* file:

```
chr1    121483982   121483983   15
chr1    121483983   121483984   12
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
```

4.6.2.2 *.interval_summary

The **.interval_summary* file is a tab delimited file which has as its first three columns the standard BED fields. The column fields that follow are statistics calculated over the interval region specified on the same record line. These are described below:

- *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 still printed to the output file. To filter out intervals with zero coverage, set *vc-emit-zero-coverage-intervals=false* in the user configuration 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       …
```

4.6.3 Coverage Analysis Examples

This section presents some useful commands for analysis of the files. For ease of readability, the user may wish to use the Linux `column -t` command on the **.interval_statistics* file and then redirect it to another file.

To filter regions based on coverage criteria, the recommended procedure is to use `awk`. Because the file is already tab delimited, the default `awk` field separator (white space) allows manipulation of the file with minimal effort. The following examples can be extended to also work on the **.genomecov.bed* file as well.

```
$ # Regions with mean coverage (fifth field $5) greater than 30x
$ awk ‘{if ($5 > 30) print $0}’ *.interval_summary
```

```
$ # Regions within chr5 (first field $1)
$ awk ‘{if ($1 == “chr5”) print $0}’ *.interval_summary
```

```
$ # Regions within chr5 (first field $1) with mean coverage (fifth field $5) greater than 30x
$ awk ‘{if (($1 == “chr5”) && ($5 > 30)) print $0}’ *.interval_summary
```

4.7 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) which better encompasses the likelihood that a call is correct or not. The VQSLOD metric gets added to the INFO field of each variant call record and can be later used 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. Details of the algorithm are further described within this section.

4.7.1 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 user 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 is generated for each pass. The negative model is generated from the worst performing sites via the *--vqsr-lod-cutoff* threshold. Once 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.

The end user can further filter out call sites at a specific threshold by specifying tranche values, which indicate the target sensitivity levels. The VQSR module will then calculate the minimum VQSLOD score required to meet this target sensitivity level. The parameter specified as *–vqsr-filter-level* will determine the threshold to filter out calls.

4.7.2 Usage and Settings

Enabling VQSR post processing adds only minutes to the DRAGEN pipeline for a whole genome analysis. The VQSLOD annotations are calculated for both SNPs and INDELs and will be output in an additional VCF file *<output-file-prefix>.vqsr.vcf*. To enable VQSR, add *--enable-vqsr true* to the DRAGEN command line.

A description of VQSR specific command line options is given below.

`enable-vqsr`

Enables the VQSR post processing module. When used in conjunction with *--enable-variant-caller* or *--enable-joint-genotyping*, the VQSR processing will occur after the variant calling stages. When used in conjuction with *--vqsr-input*, The VQSR engine works in standalone mode to process VCFs.

`vqsr-config`

Path to the VQSR configuration parameters. DRAGEN can optionally read in a configuration file which stores the VQSR parameters. VQSR parameters that are specified in the configuration file may be overridden with options specified on the command line. The configuration file is useful if the user wishes to store settings that are used across multiple runs, such as the tranche settings or resource files. An example configuration file is given as *dragen-VQSR.cfg* under /opt/edico/config.

vqsr-input

The VQSR input VCF to be processed. If this option is specified on the DRAGEN command line, then VQSR will run in standalone mode. This allows the user to run VQSR on VCFs that were generated ahead of a time.

vqsr-annotation

The annotations to be used for building the models. The format of this option is a comma separated string: the mode of operation followed by a list of annotations to be used under that mode: <mode>,<annotation>,<annotation>…

The <mode> parameter is either *SNP* or *INDEL*. If only *SNP* is specified, then only SNPs will be processed with VQSR. If only *INDEL* is specified, then only INDELs will be processed. If both *SNP* and *INDEL* are specified, via calling the option twice, then both SNPs and INDELs will be processed.

The list of annotations come from the INFO column of a VCF. The user may specify up to 8 annotations. These annotations and their associated values will be used to build the model.

Below is an example of the VQSR annotations that may be specified.

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

vqsr-resource

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. Note that 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: the mode of operation, the prior value to weight this resource, then the path the resource file: <mode>,<prior>,<resource file>.

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

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

The <resource file> is the path and filename of the VCF file to be used as the resource file.

Below is an example of how to specify the VQSR resource files:

```
--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

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 used will be 100.0, 99.99, 99.90, 99.0, and 90.0.

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

vqsr-filter-level

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

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

vqsr-lod-cutoff

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

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

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

output-directory

Similar to other DRAGEN operating modes, the output directory where output files will be stored is specified via --*output-directory*.

output-file-prefix

Similar to other DRAGEN operating modes, the output file prefix to name all output files is specified via --*output-file-prefix*.

4.7.3 Example

The VQSR module can be run as a standalone tool, in the case that the user already has a VCF which needs annotating and filtering. Here is an example stdout:

```
==================================================================
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 0.160654
Cluster 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 algorithm
K-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
```

5 DRAGEN^{TM} Transcriptome Pipeline

The DRAGEN^{TM} Transcriptome Pipeline utilizes the DRAGEN^{TM} RNA-seq spliced aligner. The majority of the functionality and options outlined in sections 3 and 4 are also applicable to RNA-seq processing. At a minimum, it is advisable to first read through sections 3 and 4.1 to 4.4 before proceeding. In this section, parameters specific to the RNA mode of operation are explained.

5.1 Mapping

The following parameters control the mapping stage of the RNA spliced aligner. Each parameter should be prefixed with Mapper.<param-name>.

5.1.1 Intron Handling

`min-intron-bases`

For RNA-seq mapping, a reference alignment gap may be interpreted as a deletion or an intron. In the absence of an annotated splice junction, the parameter *min-intron-bases* 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 may be returned with annotated splice junctions shorter than this threshold.

`max-intron-bases`

The *max-intron-bases* parameter controls the largest possible intron that is reported. This is useful for preventing false splice junctions which would otherwise be reported. Set this parameter to a value that is suitable to the species you are mapping against.

`ann-sj-max-indel`

For RNA-seq, seed mapping may discover a reference gap in the position of an annotated intron, but with slightly different length. If the length difference doesn't exceed this parameter, the mapper will investigate 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 parameter and very near annotated splice junctions are not likely to be detected; higher values may increase mapping time and false detections.

5.2 Aligning

The following parameters control the aligner stage of the RNA spliced aligner. Each parameter should be prefixed with Aligner.<param-name>.

5.2.1 Smith-Waterman Alignment Scoring Settings

Refer to Section 4.2.1 for more details about the alignment algorithm used within DRAGEN^{TM}. The following scoring parameters are specific to the processing of canonical and non-canonical motifs within introns.

`intron-motif12-pen`

The *intron-motif12-pen* parameter 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).

`intron-motif34-pen`

The *intron-motif34-pen* parameter 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).

`intron-motif56-pen`

The *intron-motif56-pen* parameter 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).

`intron-motif0-pen`

The *intron-motif0-pen* parameter controls the penalty for non-canonical motifs. The default value calculated by the host software is 6*(match-score + mismatch-pen).

5.2.2 Compatibility with Cufflinks

Cufflinks may require spliced alignments to emit the XS:A strand tag. This tag will be present in the SAM record if the alignment contains a splice junction. The values which this tag may take are defined as follows: '.' (undefined), '+' (forward strand), '-' (reverse strand), or '*' (ambiguous).

If the spliced alignment has an undefined strand or a conflicting strand, then the alignment may be suppressed with the *no-ambig-strand* parameter set to 1.

Cufflinks also expects that the MAPQ for a uniquely mapped read be a single value. This can be controlled with the host software parameter *--rna-mapq-unique.* Setting this parameter to a non-zero value forces all uniquely mapped reads to have a MAPQ equal to this value.

5.2.3 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 will impact the MAPQ estimate. These adjustments are outlined in section 4.2.1 and 5.2.1.

An additional parameter specific to RNA is *--mapq-strict-sjs*. This parameter applies where at least one exon segment is aligned confidently, but there is ambiguity about some possible splice junction. When this parameter is 0, a higher MAPQ value is returned, expressing confidence that the alignment is at least partially correct. When this parameter is 1, a lower MAPQ value is returned, expressing the splice junction ambiguity.

Some downstream tools such as Cufflinks expects the MAPQ value to be a unique value for all uniquely mapped reads. This can be controlled with the host software parameter *--rna-mapq-unique.* Setting this parameter to a non-zero value overrides all MAPQ estimates based on alignment score. Instead, all uniquely mapped reads will have a MAPQ set to equal to the value of *rna-mapq-unique*. All multi-mapped reads will 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.

5.3 Input Files

In addition to the standard input files (*.fastq, *.bam, etc), DRAGEN^{TM} may additionally take in a gene annotations file which will aid in the detection of splice junctions during RNA-seq mapping and aligning.

5.3.1 Gene Annotation File

To specify a gene annotation file, use the *-a* (*--ann-sj-file*) command line option. The input file must conform to the GTF/GFF/GFF3 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 GFF3 file may be used. Each 'exon' feature must have as a 'Parent' a transcript identifier which will be used to group exons. An example of a valid GFF3 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^{TM} host software parses the file for exons within the transcripts and produces splice junctions. The following print will display 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 which 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 which discards any splice junctions does 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 host software parameter *--rna-ann-sj-min-len*, which has a default value of 6.

5.3.2 Two-Pass Mode

In addition to reading in GTF/GFF/GFF3 files for use as annotated splice junctions, the host software is also capable of reading in a SJ.out.tab file (see section 5.4.2 for details on this file produced by DRAGEN^{TM}). This enables DRAGEN^{TM} to run in a "2-pass" mode, where the SJ.out.tab denoting the splice junctions discovered in the first pass are used to guide the mapping and alignment of a second invocation of DRAGEN^{TM}. This mode of operation is useful when a gene annotations file is not readily available for your dataset.

5.4 Output Files

The output files generated when running DRAGEN^{TM} in RNA mode is similar to the DNA mode, but it additionally produces extra information related to spliced alignments. Details regarding the splice junctions are present both in the SAM alignment record as well as an additional file, the SJ.out.tab file.

5.4.1 Alignments in BAM/SAM

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

5.4.1.1 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. More details are in section 5.2.2.

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 – This is 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 – This is 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. It's value ranges from 1 … NH. This tag may be used for downstream tools such as featureCounts.

5.4.2 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

1. contig name
2. first base of the splice junction (1-based)
3. last base of the splice junction (1-based)
4. strand (0: undefined, 1: +, 2: -)
5. intron motif: 0: non-canonical, 1: GT/AG, 2: CT/AC, 3: GC/AG, 4: CT/GC, 5: AT/AC, 6: GT/AT
6. 0: unannotated, 1: annotated, only if an input gene annotations file was used
7. number of uniquely mapping reads spanning the splice junction
8. number of multi-mapping reads spanning the splice junction
9. maximum spliced alignment overhang

The "maximum spliced alignment overhang" (column 9) field in the SJ.out.tab file is the anchoring alignment overhang. For example, if a read is spliced as ACGTACGT------------ACGT, then the overhang is 4. For the same splice junction, across all reads that span this junction, the maximum overhang is reported. The maximum overhang is a confidence indicator that the splice junction is correct based on anchoring alignments.

There are two SJ.out.tab files generated by the DRAGEN™ host software, an unfiltered version and a filtered version. The records in the unfiltered file is 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 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 non-canonical 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 non-canonical motif and the maximum spliced alignment overhang is < 30.
- SJ is a canonical motif and the maximum spliced alignment overhang is < 12.

We recommend that you use the filtered SJ.out.tab for any downstream analysis or post processing tools. Alternatively, the user may wish to use the unfiltered SJ.out.tab and apply their own filters (for example, with some basic awk commands).

Note that the aforementioned filter does not apply to the alignments present in the BAM or SAM file.

5.4.3 Chimeric.out.junction

If there are chimeric alignments present in the sample, then a supplementary Chimeric.out.junction file will also be output. This file has information about split-reads which can be used to perform downstream gene fusion detection. Each line contains one chimerically aligned read. The columns of the file is 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 a 1
8. N/A – not used, but is present to be compatible with other tools. It will always be a *
9. N/A – not used, but is present to be compatible with other tools. It will always be a *
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** which is encoded with the operation **p**. For paired end reads, the sequence of the second mate is always reverse complemented before determining strandedness.

Below is an example entry which 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
```

5.5 Gene Fusion Detection

The DRAGEN™ Gene Fusion module leverages 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 in case the end user wishes to perform their own filtering.

5.5.1 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, simply add the command line option *–enable-rna-gene-fusion=true* to your current RNA-seq command line scripts. Note that the DRAGEN™ Gene Fusion module requires the use of a gene annotations file in the format of a GTF or GFF file.

A full example command line for running an end to end RNA-seq experiment is shown below:

```
/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 will be printed.

```
=================================================================
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
=================================================================
ann-sj-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
```

5.5.2 Gene Fusion Standalone

The DRAGEN™ Gene Fusion module can be run as a standalone utility, taking in the **.Chimeric.out.junction* file and the gene annotations file as a GTF/GFF file. This is useful if the user wishes to try out various configuration parameters 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, simply pass in the already generated **.Chimeric.out.junction* file as a parameter to *--rna-gf-input*.

An example command line for running a the gene fusion module as a standalone utility is shown below:

```
/opt/edico/bin/dragen \
 -a $GTF \
 --rna-gf-input $INPUT_CHIMERIC \
 --output-dir $OUTPUT \
 --output-file-prefix $PREFIX \
 --enable-rna true \
 --enable-rna-gene-fusion true \
```

5.5.3 Gene Fusion Candidates

The detected gene fusion events are listed in the fusion_candidate output files. Two files will reside in the output directory, **.fusion_candidates.preliminary* and **.fusion_candidates.final*. The preliminary file is a pre-filtered list of candidate events detected, while the final file has the candidate events with a sufficiently high confidence after passing all filters (described below). The files contain 5 columns listing simply the fusion gene, a score, the two breakpoints, and the supporting reads. 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:-
```

5.5.4 Gene Fusion Parameters and Filters

Various filters are implemented to help mitigate the number of false positive gene fusion candidates. The following thresholds and parameters are user configurable. These filters will be applied and all candidates which still qualify will be emitted in the **.fusion_candidates.final* output file. For all pre-filtered fusion candidates, see the **.fusion_candidates.preliminary* output file.

`rna-gf-blast-pairs`

A file listing gene pairs which have high level of similarity. This list of gene pairs will be used as a homology filter to reduce false positives. One method to generate this file is to follow the instructions as described from FusionFilter: https://github.com/FusionFilter/FusionFilter/wiki/Building-a-Custom-FusionFilter-Dataset

`rna-gf-restrict-genes`

When parsing the gene annotations file (GTF/GFF) for use in DRAGEN^{TM} Gene Fusion, 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 setting is true.

`rna-gf-min-support`

Minimum number of supporting reads to call a gene fusion event. The default value is 2.

`rna-gf-min-support-be`

Minimum number of supporting reads to call a double broken exon gene fusion event. The default value is 10.

`rna-gf-min-anchor`

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`

Minimum distance to neighboring fusion events for them to be considered unique. If two candidate breakpoints are within this distance, then they will be 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, with respect to 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`

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.

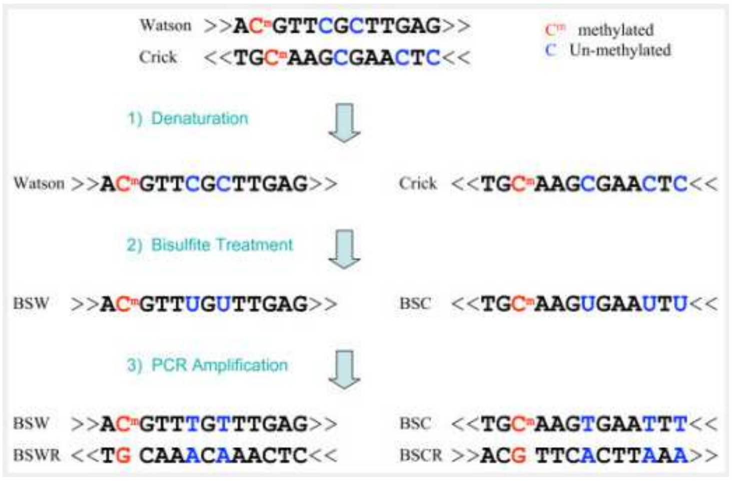
`rna-gf-exon-snap`

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 is greater than this thres

6 DRAGEN EP - Epigenome Pipeline

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 non-directional or directional. For non-directional, each double-stranded DNA fragment will yield four distinct strands for sequencing, post-amplification, as shown in the figure below:



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 will be 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 bisfulfite 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, one can determine the best alignment and most likely strand type for each read or read pair. This information is required for downstream methylation calling.

6.1 DRAGEN Automation of Methylation Alignments

DRAGEN™ currently does not perform methylation calling itself; however it can generate a separate BAM file for each of the constraints and conversions required by directional, non-directional, and directional-complement protocols. You can then feed these BAM files into a third party tool for methylation calling. At Edico, we have performed this analysis with a version of Bismark that we modified to read from DRAGEN™ BAM files instead of having Bismark internally run Bowtie or Bowtie2. We can provide instructions for implementing this solution upon request.

You can specify the protocol using the *--methylation-protocol* configuration variable. These settings cause DRAGEN™ to output a series of BAM files, each of which is run with a different set of 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. The following table describes these runs:

Protocol	BAM #	Reference	Read 1	Read 2 (if paired)	Orientation constraint
directional					
	1	C->T	C->T	G->A	forward-only
	2	G->A	C->T	G->A	RC-only
non-directional, 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, we perform alignment runs with the two combinations of base conversions and orientation constraints that correspond to these strands (directional runs 1 & 2 above). However, with **non-directional** protocols, reads from each of the four strands are equally likely, so we must also perform alignment runs with two more combinations of base conversions and orientation constraints (non-directional runs 3 & 4 above).

The **directional-complement** protocol is similar to the directional protocol in purpose, except sequencing reads are only derived from the reverse-complement strands of the BSW and BSC strands. Bismark lacks a directional-complement mode, so we must use its non-directional mode, which requires all four BAM files. For the directional-complement protocol, we expect very few good alignments from runs 1 and 2, so DRAGEN is automatically tuned to a faster analysis mode for those runs. Bismark can process the resulting four DRAGEN BAM files in its non-directional mode.

For any protocol, you must run with a reference generated with *--ht-methylated* enabled, as described in Section 7.3, Pipeline Specific Hash Tables. Then a single DRAGEN™ run will produce 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 will include MD tags, and will have "/1" or "/2" appended to the names of paired-end reads for Bismark-compatibility. These BAM files follow these 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 variables,

CT and GA correspond to the base conversions listed in the table above.

This is an example dragen command line for the directional protocol:

```
dragen --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
```

7 Preparing a Reference Genome for Use with DRAGEN™

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 Edico hardware. The parameters used in this pre-processing step offer tradeoffs between performance and mapping quality. In this Section, we discuss these configuration choices in detail.

Please note that your DRAGEN™ system is shipped with reference genomes hg19 and GRCh37 both pre-installed 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 Edico-supplied references. 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 pre-processing parameters.

7.1 DRAGEN™ Hash Table Background

The DRAGEN™ mapper extracts many overlapping seeds (subsequences or K-mers) from each read, and looks those seeds up 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.

7.1.1 Reference Seed Interval

The size of the DRAGEN^{TM} 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, i.e., roughly 3 billion seeds from a human genome. This requires at least 32GB of memory on the DRAGEN^{TM} PCIe board. To operate on larger, non-human 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; e.g. *"--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.

7.1.2 Hash Table Occupancy

It is characteristic of hash tables that they are allocated to a certain size, but internally 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^{TM} hash table; approximately 90% occupancy is a good upper bound. A little 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 also become increasingly slow to query for some seeds for which the DRAGEN^{TM} mapper is looking.

7.1.3 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 8GB to 31.5GB 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, since there are more seed positions available to avoid deviations. On balance, *"--ht-seed-len 21"* is recommended for 100bp to 150bp reads, *"--ht-seed-len 17"* to *"--ht-seed-len 19"* for short reads (e.g. 36bp), and *"--ht-seed-len 27"* for long reads (250bp+).

7.1.4 Hash Table / Seed Extensions

Due to repetitive sequences, some seeds of any given length will match many locations in the reference genome. DRAGEN^{TM} 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"* argument; for example, for short reads, it is advisable to set the maximum extended seed shorter than the read length, since extensions longer than the whole read can never match.

It is also possible to tune how aggressively seeds are extended using the following options, but this is advanced usage:

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

Basically, 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 (shown below), lean aggressively toward mapping accuracy, with relatively short seed extensions and high hit frequencies.

--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

7.1.5 Seed Frequency Limit and Target

One primary or extended seed may certainly match multiple places in the reference genome. All such matches are populated into the hash table, and retrieved when the DRAGEN^{TM} 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"* argument. 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^{TM} mapping quality notably, for two reasons. First, since 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 some other seed position is unique enough to return one or more matches, the read can still be properly mapped. On the other hand, if all seed positions were rejected as high frequency, often this means 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 *"--ht-max-seed-freq 16"* 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, since correspondingly large numbers of possible mappings will be occasionally considered.

In addition to a frequency limit, a *target* seed frequency can be specified with option *"--ht-target-seed-freq"*. This target frequency comes into play 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 *"--ht-target-seed-freq 4"* means that the software will be biased toward generating shorter seed extensions than necessary to map seeds uniquely.

7.2 Command Line Options

DRAGEN*'s --build-hash-table* is the option you use 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 pre-existing 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, although some padding is automatically inserted between reference sequences. For example, hg19 has 3,137,161,264 bases in 93 sequences; this is encoded in 1,568,841,408 bytes = 1.46GB, where 1GB means 1GiB or 2^{30} bytes.
hash_table.bin	Hash table for the DRAGEN™ mapper to look up primary seeds with length specified by option --ht-seed-len and extended seeds of various lengths.
hash_table.cfg	A list of parameters and attributes for the generated hash table, in a text format. You can inspect this file to recall key information about the reference genome and hash table, but do not edit it, because other tools use it as input to configure the DRAGEN™ hardware.
hash_table_stats.txt	A text file listing extensive internal statistics on the constructed hash table. This is purely for information purposes; it is not consumed by other tools. Here you can see, for example, the hash table occupancy percentages.

The options for building the hash table are as follows:

Usage:

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

Options:

```
--build-hash-table                  arg   Generate a reference/hash table
--output-directory                  arg   Pre-existing directory for hash table output files
--ht-reference                      arg   Reference in FASTA format
--ht-ref-seed-interval              arg   Number of positions per reference seed
--ht-seed-len                       arg   Initial seed length to store in hash table
--ht-max-ext-seed-len               arg   Maximum extended seed length
--ht-max-seed-freq                  arg   Maximum allowed frequency for a seed match after
                                          extension attempts
--ht-target-seed-freq               arg   Target seed frequency for seed extension
--ht-soft-seed-freq-cap             arg   Soft seed frequency cap for thinning
--ht-max-dec-factor                 arg   Maximum decimation factor for seed thinning
--ht-cost-coeff-seed-len            arg   Cost coefficient of extended seed length
--ht-cost-coeff-seed-freq           arg   Cost coefficient of extended seed frequency
--ht-cost-penalty                   arg   Cost penalty to extend a seed by any number of bases
--ht-cost-penalty-incr              arg   Cost penalty to incrementally extend a seed another step
--ht-rand-hit-hifreq                arg   Include a random hit with each HIFREQ record
--ht-rand-hit-extend                arg   Include a random hit with each EXTEND record of this
                                          frequency
--ht-num-threads                    arg   Worker threads for generating hash table
--ht-size arg                       arg   Size of hash table, units B|KB|MB|GB
--ht-mem-limit                      arg   Memory limit (hash table + reference), units B|KB|MB|GB
```

These options are discussed further in the sections below.

7.2.1 Input/Output Options

The *"--ht-reference*" and *"--output-directory"* arguments are required for building a hash table. The *"--ht-reference*" option specifies the path to the reference FASTA file, while *"--output-directory"* specifies a pre-existing directory where the hash table output files will be written. We recommend organizing various hash table builds into different folders. As a best practice, folder names should include any non-default parameter settings used to generate the contained hash table.

7.2.2 Primary Seed Length (--ht-seed-len)

This parameter specifies the initial length in nucleotides of seeds from the reference genome to populate into the hash table. At runtime, the mapper will extract seeds of this same length from each read, and look 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 16GB to 64GB, covering typical sizes for whole human genome, or k=26 for sizes from 4GB to 16GB.

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 will typically fail 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(3.1G) ≈ 16, so it requires at least 16 choices from 4 nucleotides to distinguish 3.1G reference positions.

7.2.2.1 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 hurt 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. Runtime quality filters such as *aln_min_score* can control the accuracy issues with very short seeds.

7.2.2.2 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”, e.g. K=16 for whole human genome.

7.2.2.3 Application Considerations

Read Length

Generally, shorter seeds are appropriate for shorter reads, and longer seeds for longer reads. Within a short read, a small number of 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. E.g. in a 36bp read, just one SNP in the middle can block seeds longer than 18bp from matching the reference. By contrast, in a 250bp read, it takes 15 SNPs to exceed a 0.01% chance of blocking even 27bp seeds.

Paired Ends

The use of paired end reads can make longer seeds yield good mapping accuracy. DRAGEN^{TM} 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 don’t match the reference well anywhere are more likely to map using short seeds to find partial matches to the reference.

7.2.3 Maximum Seed Length (--ht-max-ext-seed-len)

This parameter limits the length of extended seeds populated into the hash table. Primary seeds (length specified by --*ht-seed-len*) that match many reference positions may 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.

7.2.3.1 When to Limit Seed Extension

Using this parameter is recommended for short reads, e.g. less than 50bp. In such cases, it is helpful to limit seed extension to the read length minus a small margin, such as 1-4bp. For example, with 36bp reads, setting *“--ht-max-ext-seed-len 35”* may be appropriate. This will ensure that the hash table builder will 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, e.g. *"--ht-max-ext-seed-len 99"* for 100bp 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.

7.2.4 Maximum Hit Frequency (--ht-max-seed-freq)

This parameter 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, then 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, then 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. For a whole human genome reference, other parameters being default, the practical minimum is 8.

7.2.4.1 Accuracy Considerations

Generally, a higher maximum hit frequency leads to more successful mapping. This is for two reasons: first, a higher limit rejects fewer reference positions that cannot map under it; and 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 (see above under *--ht-seed-len*), 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.

7.2.4.2 Speed Considerations

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

7.2.5 Program Options

7.2.5.1 Threads (--ht-num-threads)

The *--ht-num-threads* parameter determines the maximum number of worker CPU threads that are used to speed up hash table construction. At most 32 threads are allowed. This parameter defaults to 8. If your server supports execution of more threads, we recommend using the maximum possible. For example, Edico Genome turnkey servers contain 24 cores that have hyper-threading enabled, so a value of 32 should be used on such a system. When using a higher value, be sure to also adjust *--ht-max-table-chunks*, as described in the next section. The turnkey servers have 128 GB of memory available.

7.2.5.2 Maximum Table Chunks in Memory (--ht-max-table-chunks)

This parameter controls the memory footprint during hash table construction, by limiting the number of ~1GB hash table chunks that reside in memory simultaneously. Each additional chunk consumes roughly twice its size (~2GB) in system memory during construction.

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

7.2.6 Size Options

7.2.6.1 Memory Limit (--ht-mem-limit)

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

7.2.6.2 Hash Table Size (--ht-size)

This parameter directly specifies the hash table size to generate, rather than letting an appropriate table size be calculated 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.

7.2.7 Seed Population Options

7.2.7.1 Seed Interval (--ht-ref-seed-interval)

This parameter defines the step size between positions of seeds in the reference genome populated into the hash table. An interval of 1 (default) means every seed position is populated, 2 means 50% of positions are populated, etc. Non-integer values are supported, e.g. 2.5 yields 40% populated.

Seeds from a whole human reference are easily 100% populated with 32GB memory on DRAGEN™ boards. The main reason to change this setting is if a substantially larger reference genome is used.

7.2.7.2 Soft Frequency Cap for Seed Thinning (--ht-soft-seed-freq-cap), Maximum Decimation Factor for Seed Thinning (--ht-max-dec-factor)

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 option *--ht-soft-seed-freq-cap* , only a fraction of seed positions are populated to stay under the cap. Option *--ht-max-dec-factor* specifies a maximum factor by which seeds can be thinned; e.g. *"--ht-max-dec-factor 3"* retains at least 1/3 of the original seeds, and *"--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, and can 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.

7.2.7.3 Random Sample Hit with HIFREQ Record (--ht-rand-hit-hifreq), Random Sample Hit with EXTEND Record (--ht-rand-hit-extend)

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 which 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.

Set *"--ht-rand-hit-hifreq 1"* to include a sample hit. The "*--ht-rand-hit-extend*" option is a minimum pre-extension hit count to include a sample hit, or zero to disable. Modifying these parameters is *not* recommended.

7.2.8 Seed Extension Control

DRAGEN^{TM} 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 some 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 parameters define that cost function.

7.2.8.1 Target Hit Frequency (--ht-target-seed-freq)

This parameter defines the ideal number of hits per seed for which seed extension should aim. Higher values lead to fewer and shorter final seed extensions, since shorter seeds tend to match more reference positions.

7.2.8.2 Cost Coefficient for Seed Length (--ht-cost-coeff-seed-len)

This parameter 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.

7.2.8.3 Cost Coefficient for Hit Frequency (--ht-cost-coeff-seed-freq)

This parameter 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.

7.2.8.4 Cost Penalty for Seed Extension (--ht-cost-penalty)

This parameter 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.

7.2.8.5 Cost Increment for Extension Step (--ht-cost-penalty-incr)

This parameter 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.

7.3 Pipeline Specific Hash Tables

By default, when building a hash table DRAGEN^{TM} will configure the parameters to work for DNA-seq processing. If your system has a Transcriptome license present, then DRAGEN^{TM} will automatically generate a hash table that is both compatible for DNA-seq and RNA-seq processing. To do an RNA-seq alignment run, you refer to the original *--output-directory*, not to the automatically generated subdirectory.

As described in section 6, DRAGEN™ methylation runs require you to build a special pair of hash tables, with reference bases converted from C->T for one table, and G->A for the other. When you run the hash table generation with the *--ht-methylated* option, these conversions are done automatically, and the converted hash tables are generated in a pair 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, make sure to 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 will probably find it necessary to tune the hash table parameters differently than you would in a conventional hash table build. As a starting point, we recommend the following settings for building these 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
```

8 Additional DRAGEN^{TM} Pipelines

The DRAGEN^{TM} platform is highly flexible and can run several different NGS data analysis pipelines. In addition to the available pipelines described previously, the DRAGEN^{TM} platform can be tailored to fit other models or use cases. One such example is its use for non-invasive prenatal testing (NIPT).

8.1 DRAGEN NP - NIPT Pipeline

Non-invasive prenatal testing (NIPT) for detecting genomic aberrations is possible via sequencing of circulating cell-free fetal DNA (ccfDNA) fragments present in maternal blood plasma. The ccfDNA fragments are typically sequenced at low coverage with very short read (<40nt) protocols. For aligning these reads, a DRAGEN configuration has been created to take advantage of shorter mapping seeds and more permissive alignment scoring parameters. The location of the default NIPT configuration file can be found at ***/opt/edico/config/dragen-NIPT.cfg***. This can be used as a template configuration file for NIPT analysis.

The ***dragen-NIPT.cfg*** file is optimized for use with a DRAGEN^{TM} hash table created with the following command line:

dragen *--build-hash-table 1 --ht-seed-len 19 --ht-max-ext-seed-len <mval> --ht-max-seed-freq 32*

where **mval** is calculated as ***read_length-1***.

9 Tools and Utilities

9.1 Illumina BCL Data Conversion

9.1.1 Overview

BCL format is the native output format of Illumina sequencers, and consists of a directory of 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 take a long time using Illumina's provided software.

Edico provides a custom implementation of the conversion software that is more performant built into DRAGEN. This capability is accessed by running DRAGEN with the parameters *–bcl-input-dir <BCL_ROOT>, --bcl-ouput-dir <DIR>,* and *--bcl-conversion-only true* together.

It is designed to output fastq files that match Illumina's bcl2fastq 1.8 for HiSeq and MiSeq sequencer models and bcl2fastq2 2.15 for HiSeq X Ten and NextSeq sequencers. Supported capabilities include demultiplexing samples by barcode with optional mismatch tolerance, adapter sequence masking or trimming with adjustable matching stringency, inclusion or exclusion of reads marked as failed, and EAMSS filtering for HiSeq and MiSeq output.

9.1.2 Command Line Options

Required parameters for BCL conversion in DRAGEN are listed below:

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

The following additional options are identical to the same option in Illumina's bcl2fastq 1.8 or bcl2fastq2 2.15 tools as named without the *bcl-* prefix (e.g. *--bcl-use-bases-mask* has the same semantics as *--use-bases-mask* in Illumina's tools):

--bcl-barcode-mismatches <barcode mismatch tolerance>	(0-2, default 0)
--bcl-adapter-sequence <fa file> (	Perform adapter sequence trimming using specified file)
--bcl-adapter-stringency <0.5 to 1.0> (	Trim adapter sequence stringency: default 0.9)
--bcl-with-failed-reads <true/false> (	Failed reads are filtered by default)
--bcl-use-bases-mask <mask> (	Determined from RunInfo.xml if unspecified)
--bcl-sample-sheet <file path> (	Path to SampleSheet.csv file, optional if in --input-dir or BaseCalls directory)
--bcl-consolidate-fastq-segments	Concatenate fastq segments to one file per read (default off)
--bcl-no-eamss <true/false> (	Disable EAMSS filtering, on by default for HiSeq & MiSeq, unsupported on HiSeqXTen & NextSeq)
Options for HiSeqXTen & NextSeq only:	
--bcl-minimum-trimmed-read-length <#> (	Reads trimmed below this # of bases become masked at that point. Default 32)
--bcl-mask-short-adapter-reads <#> (	Reads trimmed below this # of bases become completely masked out. Default 10)

The following additional parameters are also supported:

```
--bcl-only-lane <lane #> (convert only the specified lane number: default convert all lanes)
--bcl-only-read <read #> (convert only the specified read number: default convert all reads)
```

The input BCL root directory must be specified, and the output directory must be specified if converting to FASTQ (rather than streaming to Map-Align stage). Note that, unlike Illumina's tool, the specified input path is not the BaseCalls directory but three levels higher, and should have the following files and directories within it, among others:

```
Config\
Data\
Logs\
runParameters.xml
RunInfo.xml
(etc)
```

DRAGEN™ will attempt to detect the sequencer type of the BCL input automatically. In order to over-ride this auto-detection, the operator may specify the sequencer type explicitly using the --seq-type option. Valid values are HiSeq, MiSeq, HiSeqXTen, and NextSeq.

DRAGEN™ will read compressed bcl.gz files directly, so decompressing these files yourself is not necessary.

The directory where output FASTQ files are to be stored is indicated by the *--bcl-output-dir* parameter. Each sample's output will be split into a number of files with names matching the Casava 1.8 convention discussed in Section 3.1.5. These files do not need to be concatenated in order to process using DRAGEN™ (see section 3.1.5 for details on automatic multi-file input processing). If you instead wish to have only one FASTQ file per SampleSheet entry instead of many segments, you can add *--bcl-consolidate-fastq-segments=true* to the command line to have DRAGEN consolidate the files at the end of the run. Note that this option is not recommended when producing uncompressed FASTQ files, as it will double conversion time and temporary disk space usage. For compressed FASTQ.GZ output, file consolidation is manageable and may be desired.

EAMSS filtering is enabled by default for HiSeq and MiSeq sequencer models to match Illumina's bcl2fastq 1.8 behavior, but it is recommended to disable it with *--no-eamss true*. EAMSS filtering is always disabled for HiSeqXTen and NextSeq inputs.

9.1.3 Limitations

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

9.2 Monitoring System Health

9.2.1 Overview

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. One major goal 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/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 used. 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"
```

Note that the -d argument is required to run the monitor as a daemon.

The *dragen_mond* command line options are as follows:

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

The current temperature of the DRAGEN^{TM} Bio-IT Processor may be displayed with the *dragen_info -t* command. Note that this command will not execute if *dragen_mond* is not running.

```
% dragen_info -t
FPGA Temperature: 42C  (Max Temp: 49C, Min Temp: 39C)
```

9.2.2 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 this happens, stop running the dragen software. Run through the following steps 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 Edico support if you are having trouble resolving the thermal alarm on your system.

9.2.3 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 non-fatal
3	SODIMM overheating	Logged as non-fatal

4	Power 0	Fatal – Aborts dragen software; system reboot required
5	Power 1	Fatal – Aborts dragen software; system reboot required
6	DRAGEN^{TM} Bio-IT Processor power	Logged as non-fatal
7	Fan 0	Logged as non-fatal
8	Fan 1	Logged as non-fatal
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^{TM} host software from running and require a system reboot. When a software overheating alarm is triggered, the monitor will look for and abort any running DRAGEN^{TM} processes. The monitor will continue to abort any new DRAGEN^{TM} processes until the temperature decreases to the minimum threshold and the hardware clears the chip status alarm. When the software overheating alarm clears, DRAGEN^{TM} jobs may resume executing.

Please contact Edico support with details from the log files if any of these alarms are triggered on your system.

9.3 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. As described above, if your input files are gzipped, DRAGEN will detect that and decompress the files automatically, and if your output is BAM then the files will automatically be compressed.

In testing at Edico, we have found that the speed of compression may vary by system, and using parallel software compression can outperform the hardware acceleration. Thus by default the hardware compression is disabled. If you wish to run your own benchmarks to test performance on your hardware, you can enable the hardware compressing by setting *--use-compression-hardware* to *true.*

As an added convenience, Edico 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* (short for "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 filenames (including those produced by a wildcard * character) are ignored.
- They may not be run at the same time as the DRAGEN™ host software.

They do not support the command line options found in gzip/gunzip (e.g., recursive, --fast/--best, --stdout).

9.4 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 Edico's server. Information includes date, duration, size (number of bases) and status of each runs, software version used, etc.

Communication to Edico server is secured by encryption. In case of communication error, the daemon will retry until the next morning. If the upload continues to fail, communication will be 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.

Uploaded information is made available on the customer portal as a way to track usage of the board against the monthly quota.

10 Troubleshooting

This Section describes the procedures for troubleshooting a DRAGEN^{TM} system malfunction. Refer to subsection 10.1 to determine if your system is hanging, and then follow the instructions in subsection 10.2 to collect diagnostic information after a crash or a hang. Once all information has been collected, follow the instructions in subsection 10.3 to reset your system (if needed).

10.1 How to Identify if the System is Hanging

If a run seems to be taking longer than it should, it may be hanging. Here are some things to try:

- Run the *top* command to find the active DRAGEN^{TM} 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.

- Use 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.

10.2 Sending Diagnostic Information to Edico Support

We are eager to hear your feedback on your DRAGEN^{TM} system, including any reports of system malfunction. In the event of a crash or hang, please run *sosreport* to collect diagnostic and configuration information:

```
sudo sosreport --batch
```

This tool will take several minutes to execute and will report the location where it has saved the diagnostic information in /tmp.

If the software is hanging, please perform the following additional steps:

1. Get the dragen software to dump its hang-diagnostic information

   ```
   kill -10 $( pidof dragen )
   ```

   This produces a file /tmp/hang_diag_<timestamp>.txt.

2. Save information on the internal state of the host software:

   ```
   pstack $( pidof dragen ) > /tmp/pstacks.txt
   ```

Please include all of these output files along with your report when you submit a support ticket on the DRAGEN^{TM} Portal http://dragen.edicogenome.com.

10.3 Resetting Your System after a Crash or Hang

If the DRAGEN^{TM} 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, you will see the following message from the host software:

```
  Running dragen_reset to reset DRAGEN Bio-IT processor and software
```

If the software is hanging, please collect diagnostic information as described in subsection 10.2 and then execute *dragen_reset* manually:

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

Any execution of *dragen_reset* will require the reference genome to be reloaded to the DRAGEN^{TM} board. The host software will automatically reload the reference on the next execution.

11 Appendix: Configuration File Parameters

11.1 Host Software Parameters

The following parameters are in the default section of the configuration file. The default section does not have a section name (e.g., [Aligner]) associated with it and can be located at the topmost portion of the configuration file. Note that some mandatory fields must be specified on the command line and are not present in configuration files.

Parameter Name	Description	Command line equivalent	Range
fastq-file1	FASTQ file to send through the DRAGEN™ pipeline (may be gzipped)	-1	
fastq-file2	Second FASTQ file with paired-end reads to send through the DRAGEN™ pipeline	-2	
tumor-fastq1	FASTQ file to send through 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 to send through the DRAGEN™ pipeline using the variant caller in somatic mode (may be gzipped)	--tumor-fastq2	
bam-input	Aligned BAM file to send through the DRAGEN™ variant caller	-b	
tumor-bam-input	Aligned BAM file to send through the DRAGEN™ variant caller in somatic mode	--tumor-bam-input	
pair-by-name	Whether to shuffle the order of BAM input records such that paired-end mates are processed together		
ref-dir	Directory with reference hash table. This reference will automatically be loaded to the DRAGEN™ card, if it is not already there	-r	
append-read-index-to-name	By default, DRAGEN™ names both mate ends of pairs the		true/false

	same. With this option enabled, it will instead append "/1" and "/2" to the two ends, respectively.		
pair-suffix-delimiter	Controls the delimiter used for append-read-index-to-name and for detecting matching pair names with BAM input		/ or . or :
num-threads	The number of processor threads to use	-n	
output-file-prefix	Output filename prefix that will be used for all files generated by the pipeline	--output-file-prefix	
output-directory	Output directory	--output-directory	
output-format	The format of the output file from the map/align stage. Valid values are "bam" (which is the default) "sam", or "dbam" (a proprietary Edico binary format)	--output-format	BAM/SAM/DBAM
RGID	Read group ID	--RGID	
RGLB	Read group library	--RGLB	
RGPL	Read group sequencing technology	--RGPL	
RGPU	Read group platform unit	--RGPU	
RGSM	Read group sample name	--RGSM	
RGCN	Read group sequencing center name	--RGCN	
RGDS	Read group description	--RGDS	
RGDT	Read group run date	--RGDT	
RGPI	Read group predicted insert size	--RGPI	
bcl-conversion-only	Perform Illumina BCL conversion to FASTQ format	--bcl-conversion-only	
bcl-input-dir	Input BCL directory for BCL conversion	--bcl-input-dir	
bcl-output-dir	Output BCL directory for BCL conversion	--bcl-output-dir	
bcl-barcode-mismatches	Barcode mismatch tolerance for BCL conversion	--bcl-barcode-mismatches	
bcl-adapter-sequence	Perform adapter sequence trimming using specified file	--bcl-adapter-sequence	

bcl-adapter-stringency	Adapter sequence stringency	--bcl-adapter-stringency	
bcl-with-failed-reads	Include reads marked as 'failed'	--bcl-with-failed-reads	
bcl-use-bases-mask	Use specified mask for reads	--bcl-use-bases-mask	
bcl-sample-sheet	Path to SampleSheet.csv file	--bcl-sample-sheet	
bcl-seq-type	Specify or override sequencer type	--bcl-seq-type	HiSeq/MiSeq/HiSeqXTen/NextSeq
bcl-no-eamss	Disable EAMSS filtering	--bcl-no-eamss	
bcl-minimum-trimmed-read-length	Reads trimmed below this number of bases become masked at that point	--bcl-minimum-trimmed-read-length	
bcl-mask-short-adapter-reads	Reads trimmed below this number of bases become complete masked out	--bcl-mask-short-adapter-reads	
bcl-only-lane	Convert only the specified lane number	--bcl-only-lane	
bcl-only-read	Convert only the specified read number	--bcl-only-read	
bcl-num-threads	Number of threads to use for BCL conversion	--bcl-num-threads	
bcl-clusters-per-pass	Number of clusters to process per pass	--bcl-clusters-per-pass	
dbsnp	Path to the dbSNP database VCF	--dbsnp	
enable-map-align-output	Enables saving the output from the map/align stage. This is default to true when only running map/align, but is default to false if running the variant caller	--enable-map-align-output	true/false
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-bam-compression	Enable to turn on compression of output BAM files.		true/false
enable-sort	Enable sorting after mapping/alignment		true/false
use-compression-hardware	Whether to use hardware acceleration for compressing output BAM files		true/false
enable-duplicate-marking	Enable the flagging of duplicate output alignment records	--enable-duplicate-marking	true/false
fastq-offset	FASTQ quality offset value	--fastq-offset	33 or 64
fastq-n-quality	Base-call quality to output for N bases. Automatically added to fastq-n-quality for all output N's.		0 to 255
generate-md-tags	Whether to generate MD tags with alignment output records. Defaults to false.		true/false
preserve-bqsr-tags	Whether to preserve input BAM file's BI and BD flags. Note this may cause problems with hard clipping.		true/false
methylation-protocol	Library protocol for methylation analysis		none / directional / non-directional / directional-complement
preserve-map-align-order	Produce output file that preserves original order of reads in the input file		true/false
remove-duplicates	If true, remove duplicate alignment records instead of just flagging them		true/false
enable-sampling	Automatically detect paired-end parameters by running a sample through the mapper/aligner		true/false
sample-size	Number of reads to sample when enable-sampling is true		

ref-sequence-filter	Output only reads mapping to this reference sequence	--ref-sequence-filter	
intermediate_-results_-dir	Directory to store intermediate results in (e.g. sort partitions)		
enable-variant-caller	Enables the variant variant caller	--enable-variant-caller	true/false
vc-reference	Reference file to be used during variant calling in *.fasta format	--vc-reference	
vc-target-bed	Target regions BED file	--vc-target-bed	
vc-depth-intervals-bed	Depth intervals BED file	--vc-depth-intervals-bed	
vc-enable-depth-of-coverage	Enable depth of coverage calculations	--vc-enable-depth-of-coverage	true/false
vc-sample-name	Variant caller sample name; analogous field to RGSM	--vc-sample-name	
vc-target-coverage	Target coverage for downsampling; default is 2000	--vc-target-coverage	
vc-min-read-qual	Minimum read quality (MAPQ) to be considered for variant calling; default is 20	--vc-min-read-qual	
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-max-reads-per-active-region	Maximum number of reads per region for downsampling; default is 1000		
vc-min-reads-per-start-pos	Minimum number of reads per start position for downsampling; default is 5	--vc-min-reads-per-start-pos	
dbsnp	Variant annotation database VCF (or .vcf.gz) file		
filter-flags-from-output	Filter output alignments with any bits set in 'val' present in the flags field.	--filter-flags-	

	Hex & decimal values accepted	from-output	
rna-mapq-unique	For compatibility with Cufflinks, enable this parameter with a non-zero value. Unique mappers will have a MAPQ set to this value. Multi-mappers will have a MAPQ of int(-10*log10(1 – 1 /NH))		0, 1 to 255
rna-ann-sj-min-len	Discard splice junctions which have length less than this value, during the generation of splice junctions from an annotations file (GTF/GFF/SJ.out.tab).		
build-hash-table	Generate a reference/hash table	--build-hash-table	true/false
ht-reference	Reference file in .fasta format to be used to build a hash table	--ht-reference	
ht-ref-seed-interval	Number of positions per reference seed	--ht-ref-seed-interval	
ht-seed-len	Initial seed length to store in hash table	--ht-seed-len	
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-target-seed-freq	Target seed frequency for seed extension	--ht-target-seed-freq	
ht-soft-seed-freq-cap	Soft seed frequency cap for thinning	--ht-soft-seed-freq-cap	
ht-max-dec-factor	Maximum decimation factor for seed thinning	--ht-max-dec-factor	
ht-cost-coeff-seed-len	Cost coefficient of extended seed length	--ht-cost-coeff-seed-len	
ht-cost-coeff-seed-freq	Cost coefficient of extended seed frequency	--ht-cost-coeff-seed-freq	
ht-cost-penalty	Cost penalty to extend a seed by any number of bases	--ht-cost-penalty	
ht-cost-penalty-incr	Cost penalty to incrementally extend a seed another step	--ht-cost-penalty-incr	

ht-rand-hit-hifreq	Include a random hit with each HIFREQ record	--ht-rand-hit-hifreq	
ht-rand-hit-extend	Include a random hit with each EXTEND record of this freq	--ht-rand-hit-extend	
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-max-ext-incr	Maximum bases to extend a seed by in one step	--ht-max-ext-incr	
ht-max-table-chunks	Maximum ~1GB thread table chunks in memory at once	--ht-max-table-chunks	
ht-size	Size of hash table, units B|KB|MB|GB.	--ht-size	
ht-mem-limit	Memory limit (hash table + reference), units B|KB|MB|GB.	--ht-mem-limit	
`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`	
`interleaved`	`Interleaved paired-end reads in single FASTQ`	`-i`	
`verbose`	`Enable verbose output from the` DRAGEN™ `program`	`-v`	
`version`	`Print the version and exit`	`-V`	

11.2 Mapper Parameters

The following parameters are in the [Mapper] section of the configuration file. Refer to subsection 4.1 Mapping, for more detailed information on these parameters.

Parameter Name	Description	Command line equivalent	Range
`seed-density`	`Requested density of seeds from reads queried in the hash table`		`0 > seed-density > 1`
`edit-mode`	`0 = No edits, 1 = Chain len test, 2 = Paired chain len test, 3 = Edit all std seeds`		`0 to 3`
`edit-seed-num`	`For edit-mode 1 or 2: Requested number of seeds per read to allow editing on`		`edit-seed-num >= 0`
`edit-read-len`	`For edit-mode 1 or 2: Read length in which to try edit-seed-num edited seeds`		`edit-read-len > 0`

edit-chain-limit	For edit-mode 1 or 2: Maximum seed chain length in a read to qualify for seed editing		edit-chain-limit >= 0
map-orientations	0=Normal, 1=No Rev Comp, 2=No Forward (paired end requires Normal)		0 to 2
min-intron-bases	Minimum reference deletion length reported as an intron		
max-intron-bases	Maximum intron length reported		
ann-sj-max-indel	Maximum indel length to expect near an annotated splice junction		0 to 63

11.3 Aligner Parameters

The following parameters are in the [Aligner] section of the configuration file. Refer to subsection 4.2 Aligning, for more detailed information on these parameters.

Parameter Name	Description	Command line equivalent	Range
match-score	Score increment for matching reference nucleotide		When global = 0, match-score > 0; When global = 1, match-score >= 0
match-n-score	(signed) Score increment for matching a reference 'N' nucleotide IUB code		-16 to 15
mismatch-pen	Score penalty for a mismatch		0 to 63
gap-open-pen	Score penalty for opening a gap (insertion or deletion)		0 to 127
gap-ext-pen	Score penalty for gap extension		0 to 15
unclip-score	Score bonus for reaching each edge of the read		0 to 127
global	If alignment is global (Needleman-Wunsch) rather than local (Smith-Waterman)		0,1
pe-orientation	Expected paired-end orientation: 0=FR, 1=RF, 2=FF		0,1,2

pe-max-penalty	Maximum pairing score penalty, for unpaired or distant ends		0 to 255
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		−2,147,483,648 to 2,147,483,647
map-orientations	Constrain orientations to accept forward-only, reverse-complement only, or any alignments		0 (any) 1 (forward only) 2 (reverse only)
mapq-max	Ceiling on reported MAPQ		0 to 255
supp-aligns	Maximum supplementary (chimeric) alignments to report per read		0 to 30
sec-aligns	Maximum secondary (suboptimal) alignments to report per read		0 to 30
sec-phred-delta	Only secondary alignments with likelihood within this Phred of the primary are reported		0 to 255

sec-aligns-hard	Set to force unmapped when not all secondary alignments can be output		0,1
supp-as-sec	If supplementary alignments should be reported with secondary flag		0,1
hard-clips	Flags for hard clipping: [0] primary, [1] supplementary, [2] secondary		3 bits
unpaired-pen	Penalty for unpaired alignments in Phred scale		0 to 255
dedup-min-qual	Minimum base quality for calculating read quality metric for deduplication		0 to 63

If you disable automatic detection of insert-length statistics via the --*enable-sampling* parameter, you must override all of the following variables to specify the statistics. This is described in section 4.2.3. These variables are all part of the [Aligner] section of the configuration file.

Parameter Name	Description	Command line equivalent	Range
pe-stat-mean-insert	Average template length		0 to 65535
pe-stat-stddev-insert	Standard deviation of template length distribution		0 to 65535
pe-stat-quartiles-insert	A comma-delimited trio of numbers specifying 25th, 50th, and 75th percentile template lengths		0 to 65535
pe-stat-mean-read-len	Average read length		0 to 65535

11.4 Variant Caller Parameters

The following parameters are in the Variant Caller section of the configuration file. Refer to subsection 4.5.2 Variant Caller Settings for more detailed information on these parameters.

Parameter Name	Description	Command line equivalent	Range

vc-hard-filter	Boolean expression for filtering variant calls. Parameters in the expression can include QD, MQ, FS, MQRankSum, and ReadPosRankSum. Default expression is: DRAGENHardSNP:snp: QD < 2.0 || MQ < 40.0 || FS > 60.0 || MQRankSum < -12.5 || ReadPosRankSum < -8.0; DRAGENHardINDEL:indel: QD < 2.0 || ReadPosRankSum < -20.0 || FS > 200.0		
vc-target-coverage	Target coverage for downsampling. Default is 2000.		
vc-min-read-qual	Minimum read quality (MAPQ) to be considered for variant calling. Default is 20.		
vc-min-base-qual	Minimum base quality to be considered for variant calling. Default is 10.		
vc-min-call-qual	Minimum variant call quality for inclusion. Default is 30.0		
vc-max-reads-per-active-region	Maximum number of reads per region for downsampling. Default is 1000.		
vc-min-reads-per-start-pos	Minimum required number of reads for each start position when downsampling. Default is 5.		