CRISPR-GRANT: A cross-platform Graphical Analysis Tool for high-throughput CRISPR-based genome editing evaluation

Huancheng Fu
(fuhuancheng@foxmail.com)

1 Introduction

CRISPR-GRANT, a stand-alone graphical CRISPR indel analysis tool, could be easily installed for multi-platform including Linux, Windows, and MacOS. CRISPR-GRANT offered a straightforward GUI by simple click-and-run for genome editing analysis of single or pooled amplicons and one-step analysis for whole-genome sequencing. Moreover, it also exhibited shorter run-time compared with tools currently available. Therefore, CRISPR-GRANT is a valuable addition to the current CRISPR toolkits that significantly lower the barrier for wet-lab researchers to conduct indel analysis from large NGS datasets.

CRISPR-GRANT requires input from raw FASTQ sequencing data and reference sequence in FASTA format. Fastp will be first used for quality check of the input data. After that, qualified reads will be mapped to the reference sequence by BWA-MEM. The resulting SAM file was then converted to BAM by samtools, which will be subsequently used for consensus and variants analysis by VarScan2. The output data contains QC reports, FASTQ of qualified reads, reference-mapping results, table of consensus and variants, and publication-quality plots for the number of modified and unmodified reads, alignment of top numbered reads to reference, and the frequency of indels at each position.

2 Download and install

The usage of CRISPR-GRANT is simple and intuitive. CRISPR-GRANT was a self-contained software. You just need to download and open it. Download the CRISPR-GRANT package from website (https://github.com/fuhuancheng/CRISPR-GRANT/releases), uncompressed it, go to the folder (**The path should not contain non-English letters, blank or other invalid characters.**) and click to open the program. The GUI would look like Fig1.

System requirements:

- Operating systems: Windows 7 or 10; macOS 10.13 or later; GNU/Linux (such as Debian, openSUSE, Ubuntu, Deepin). The program had been tested on Windows 7, Windows 10, macOS 10.13 (High Sierra), Debian 10, openSUSE, Ubuntu 16.04 and Deepin linux 20.
- CPU: x86_64 CPUs (64 bit).
- RAM: at least 4 GB RAM, 8 GB is recommended. For very large or whole genome sequencing (WGS) dataset, larger RAM may be necessary.
- Disk usage: it depends on the sequencing data size. For WGS data of mouse (*Mus musculus*), as large as 500GB disk space would be required for one sample.

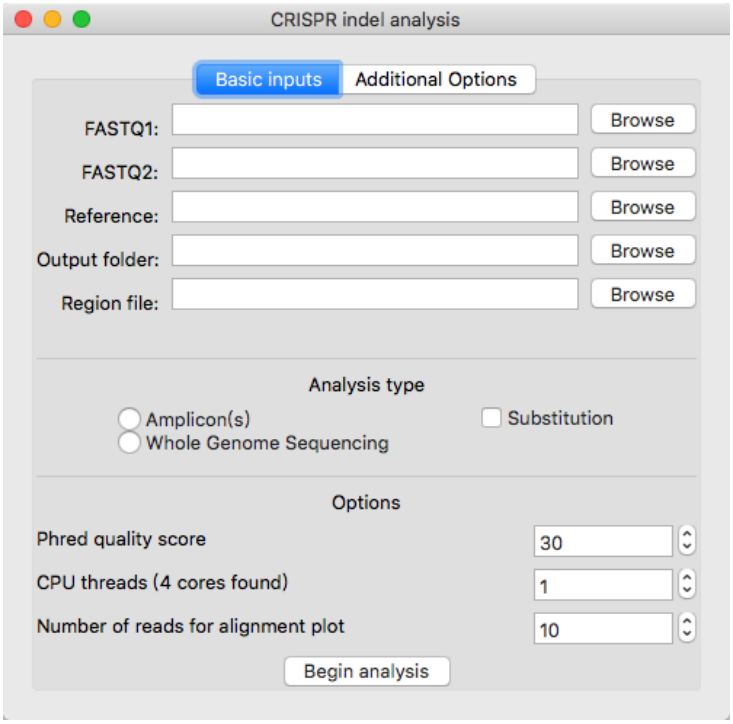

Figure 1: GUI example

3 Usage for amplicon or pooled amplicon analysis

Required files or inputs for analysis:

- FASTQ file(s) from single-end or paired-end sequencing.
- Reference sequence in FASTA file format.
- Output folder for analysis results.

Analysis for amplicon or pooled amplicons includes steps (Fig2):

1. Press "Browse" button to select FASTQ file 1 and 2. (**The file path should not contain non-English letters, blank or other invalid characters. Same to reference file and output folder, etc below.**)
2. Press "Browse" button to select reference file in FASTA format.
3. Press "Browse" button to select or input output folder name for result files.
4. Choose "Amplicon(s)" in "Analysis type" field. Check "Substitution" if you want to plot numbers of base substitution.
5. Choose additional parameters if needed, or just leave it as default.
6. Press "Begin analysis" button to begin analysis.
7. When completed, go to the result folder to check the result files.

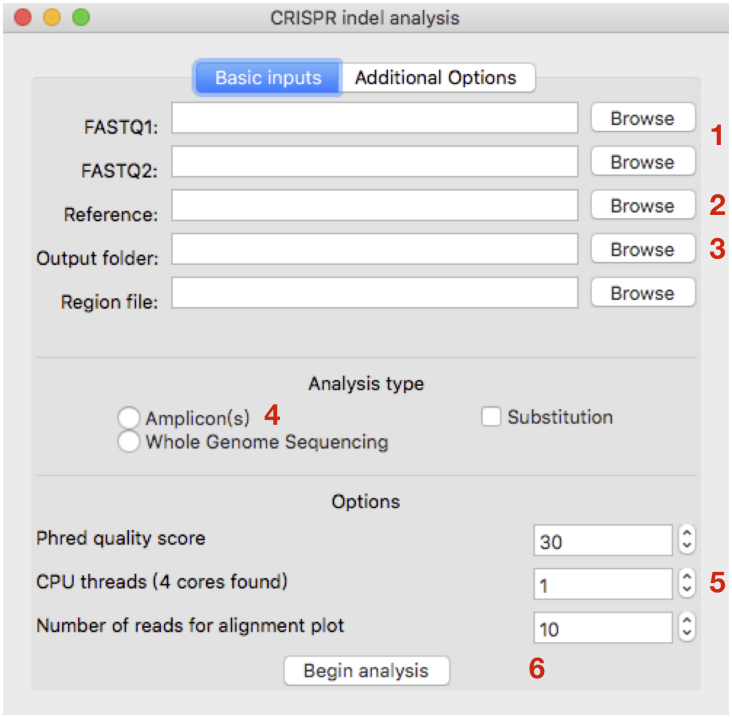

Figure 2: Steps for amplicon(s) analysis

4 Usage for WGS (whole genome sequencing) analysis

Required inputs for analysis:

- FASTQ file(s) from single-end or paired-end sequencing.
- Reference sequence in FASTA file format.
- Output folder for analysis results.
- A file containing specific regions. Format: chrome[:start[-end]], for example, chr1:42-2020, one region per line.

Since the reference genomes are generally huge, the genome reference files were not provided with the installer packages. However, it's easy to download genome sequence for a variety of organisms from UCSC (`https://hgdownload.soe.ucsc.edu/downloads.html`) or NCBI (`https://www.ncbi.nlm.nih.gov/genome/`). The downloaded genome sequence should be in FASTA format (see section4.1.2).

Analysis for WGS include steps (Fig3):

1. Press "Browse" button to select FASTQ file 1 and 2.
2. Press "Browse" button to select reference file in FASTA format.
3. Press "Browse" button to select folder for result files.
4. Press "Browse" button to select region file.
5. Choose "Whole Genome Sequencing" in "Analysis type" field.
6. Choose additional parameters if needed, or just leave it as default.

7. Press "Begin analysis" button to begin analysis.

8. When completed, go to the result folder to check the result files.

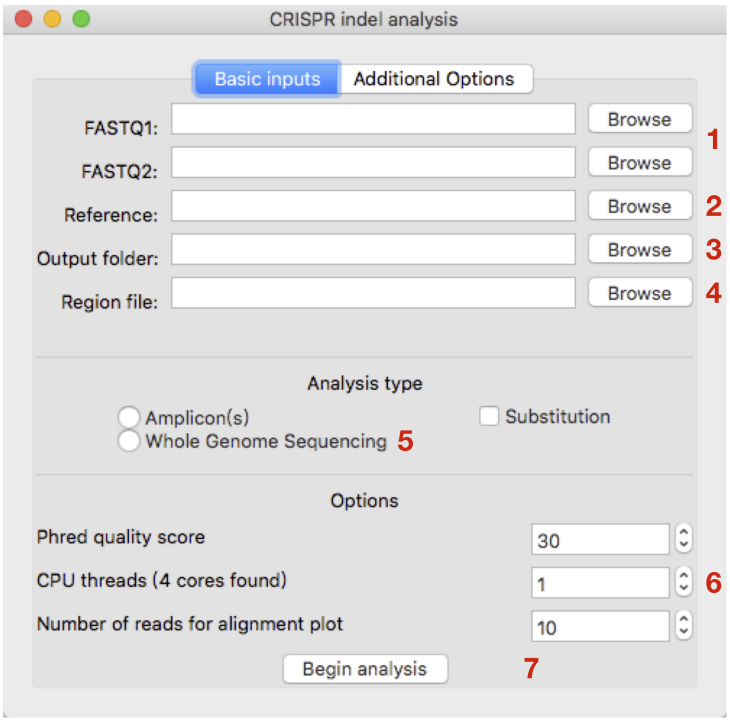

Figure 3: Steps for WGS analysis

4.1 Input file format

4.1.1 FASTQ: sequencing file format

FASTQ files (*.fastq) or compressed FASTQ files (*.fastq.gz) were both accepted.

4.1.2 FASTA: reference file format

The reference file must be in FASTA file format, file extension is arbitrary. The detailed FASTA file format explaination could be found from https://en.wikipedia.org/wiki/Fasta_format.

A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The definition line (defline) is distinguished from the sequence data by a greater-than (¿) symbol at the beginning. The word following the "¿" symbol is the identifier of the sequence, and the rest of the line is the description (optional). There should be **NO** space between the "¿" and the first letter of the identifier. It is recommended that all lines of text be shorter than or equal to 80 characters in length. Below is an example sequence in FASTA format:

```
>Sequence-name1
ATGCACTGAGGCACGGCAGGCCCAGAGCATCTCACCTGAAGCACCCTTCTTGCCTAAATCCAGCTTTCTG
TCACACTCTCCCAGAAGGAGGGGAGAGGGGGTAAAAAAATGCTGCACTGTGCGGCG
>Sequence-name2
GCGACGCGGAGCCAATCAGCGCCCGCCGTTCCGAAAGTTGCCTTTTATGGCTCGAGTGGCCGCTGTGGCG
TCCTATAAAACCCGGCGGCGCAACGC...
```

In cases such as reference for amplicon pools, all the sequence names in the given reference file should **NOT** be in duplicate.

4.2 Result files

The resulting output files include top numbered reads alignment to reference, distribution of reads counts (total reads, mapped reads, modified and un-modified reads), frequency of indels at each position along the reference, etc (Table1).

Table 1: Descriptions of output files

Numbering	File name	Format	Description
0	QC-report	html	Quality control report of FASTQ files
1	readsCount	pdf, txt	Number of reads types
2	Top_sequences_alignment	fasta, pdf	Sequence alignment of top reads to reference
3	varFreq	csv, pdf	Frequency of indels at each position along reference
4	Mapping_sorted	BAM	Sorted BAM file mapping to reference

4.2.1 QC (quality control) report

The QC report was generated by fastp (https://github.com/OpenGene/fastp). Quality profiling for both before and after filtering of reads, such as number of total reads, percentage of qualified reads, were displayed in QC report.

4.2.2 Alignment of top counts

In the result folder, the alignment of top count sequences will be given, as shown in fig4.

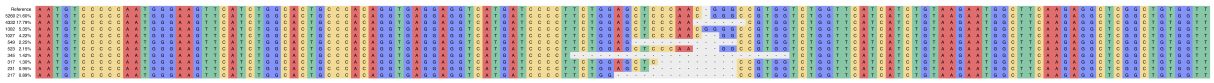

Figure 4: Example of alignment plot

4.2.3 Indel frequency along sequence

In the result folder, the plot of indel frequency along sequence will also be given, as shown in Figure5.

4.2.4 Use IGV to visualize mapping result

The resulting file of reads mapping to reference was saved to "Mapping_sorted.bam" file in BAM format (Table 1). The BAM file could be visualized using IGV (http://igv.org/) (Figure6).

5 Command line usage

CRISPR-GRANT also provides command line usage for some situations when needed.

```
indel_analysis -1=FASTQ1/file/path -2=FASTQ2/file/path -r=reference/file/path
-o=output/file/path -t=CPU_cores
```

Command "indel_analysis -h" would give more detailed help usage information.

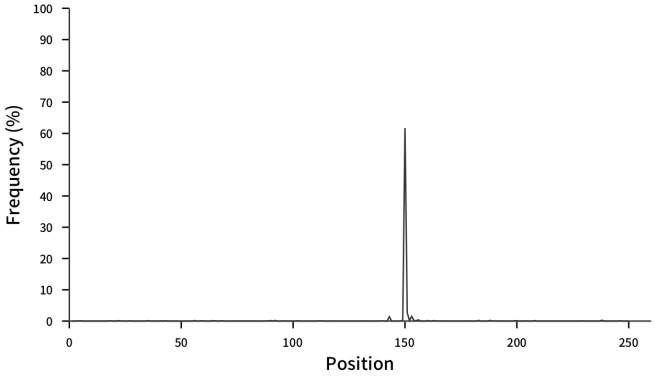

Figure 5: Example of indel frequency along sequence

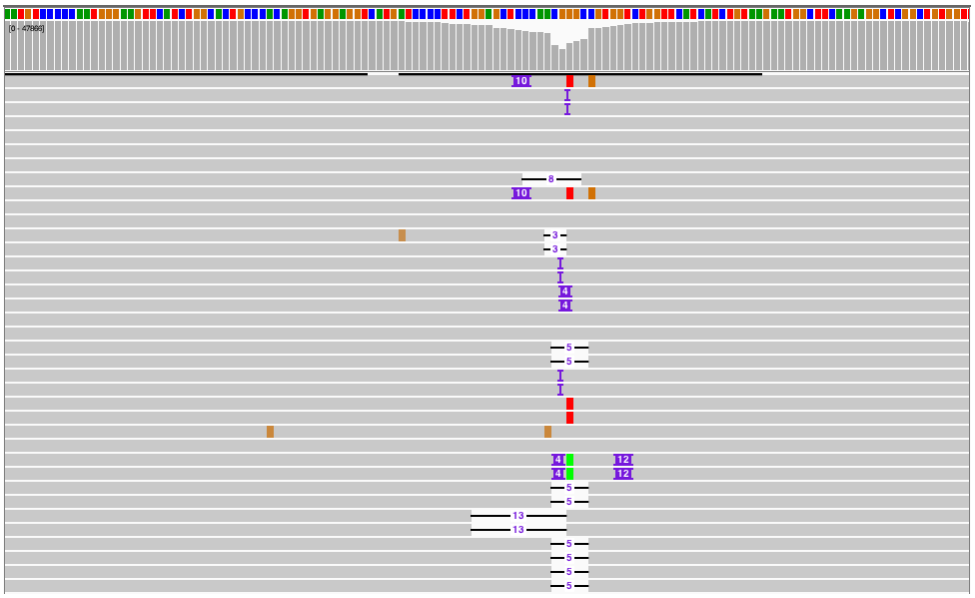

Figure 6: Use IGV to visualize BAM files

6 Compilation

Requirements:

- gcc or clang compiler. On Windows, Mingw-w64 should be installed and used for compilation.
- Nim (greater than Version 1.2.0). Please consult https://nim-lang.org for installation.
- ggplotnim (greater than version 0.3.18, https://github.com/Vindaar/ggplotnim). ggplotnim could be installed via `nimble install ggplotnim`.
- ui (greater than version 0.9.4, https://github.com/nim-lang/ui). ui could be installed via `nimble install ui`.

Download the source code, uncompressed it and change into the source code directory: `cd CRISPR-GRANT`.

On Linux compile with: `make linux`. On Mac: `make mac`. On Windows: `make windows`.

You should also have executable fastp, flash, bwa, samtools and VarScan2 binary files installed in bin folder.

The resulting bin folder should be like below:

```
indel_call_ui
bin/
|-- bwa/
|-- fastp/
|-- flash/
|-- jre11/
|-- mafft/
|-- samtools/
|-- varscan2/
|-- csv_to_fasta
|-- fasta_to_plot
|-- indel_analysis
|-- reads_count_plot
|-- sam_count
|-- snp_plot
|-- varToCsv
|-- var_plot
|-- wgsSubRegion
```

7 Get help

If you have any questions or encountered bugs when using CRISPR-GRANT, please feel free to contact us by email (fuhuancheng@foxmail.com), or creating issues at https://github.com/fuhuancheng/CRISPR-GRANT.

8 License

CRISPR-GRANT is distributed under license of GPLv3 LICENSE except the tools internally used licensed originally. Licenses of tools and libraries used within CRISPR-GRANT are listed below. Source codes of tools licensed under GPLv3 are provided within each tool's subfolder.

Table 2: Licenses of tools and libraries used

Tools	License
fastp	MIT
FLASH	GPLv3
bwa	GPLv3
samtools	MIT
VarScan2	Non-Profit Open Software License
ggplotnim	MIT
ui	MIT

Additional options, if necessary, can be added and passed directly to internally used tools in Additional Options tab (Fig7).

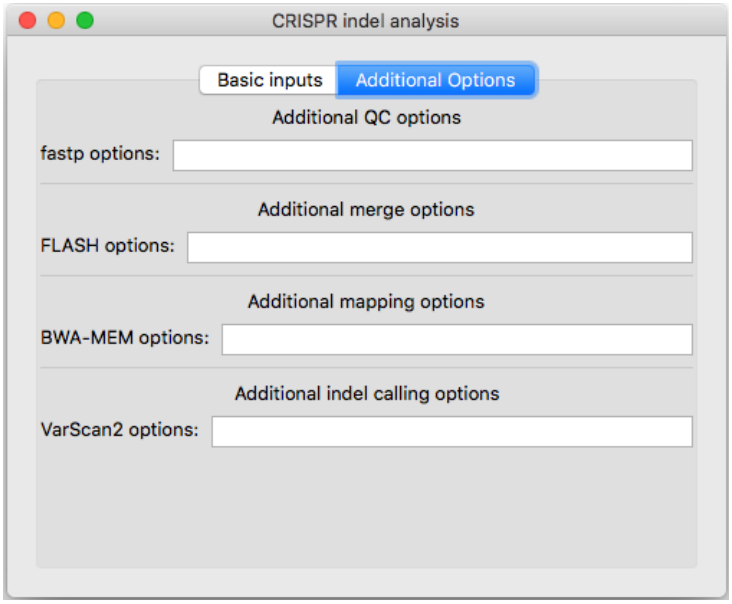

Figure 7: GUI of additional options

A Additional parameters for FLASH

Here are all available additional options for flash. Detailed additional parameters for FLASH can be accessed by execute command "flash -h" in terminal.

```
-m, --min-overlap=NUM   The minimum required overlap length between two
                        reads to provide a confident overlap.  Default:
                        10bp.

-M, --max-overlap=NUM   Maximum overlap length expected in approximately
                        90% of read pairs.  It is by default set to 65bp,
                        which works well for 100bp reads generated from a
                        180bp library, assuming a normal distribution of
                        fragment lengths.  Overlaps longer than the maximum
                        overlap parameter are still considered as good
                        overlaps, but the mismatch density (explained below)
                        is calculated over the first max_overlap bases in
                        the overlapped region rather than the entire
                        overlap.  Default: 65bp, or calculated from the
                        specified read length, fragment length, and fragment
                        length standard deviation.

-x, --max-mismatch-density=NUM
                        Maximum allowed ratio between the number of
                        mismatched base pairs and the overlap length.
                        Two reads will not be combined with a given overlap
                        if that overlap results in a mismatched base density
                        higher than this value.  Note: Any occurence of an
```

```
                    'N' in either read is ignored and not counted
                    towards the mismatches or overlap length.  Our
                    experimental results suggest that higher values of
                    the maximum mismatch density yield larger
                    numbers of correctly merged read pairs but at
                    the expense of higher numbers of incorrectly
                    merged read pairs.  Default: 0.25.

-p, --phred-offset=OFFSET
                    The smallest ASCII value of the characters used to
                    represent quality values of bases in FASTQ files.
                    It should be set to either 33, which corresponds
                    to the later Illumina platforms and Sanger
                    platforms, or 64, which corresponds to the
                    earlier Illumina platforms.  Default: 33.

-r, --read-len=LEN
-f, --fragment-len=LEN
-s, --fragment-len-stddev=LEN
                    Average read length, fragment length, and fragment
                    standard deviation.  These are convenience parameters
                    only, as they are only used for calculating the
                    maximum overlap (--max-overlap) parameter.
                    The maximum overlap is calculated as the overlap of
                    average-length reads from an average-size fragment
                    plus 2.5 times the fragment length standard
                    deviation.  The default values are -r 100, -f 180,
                    and -s 18, so this works out to a maximum overlap of
                    65 bp.  If --max-overlap is specified, then the
                    specified value overrides the calculated value.

                    If you do not know the standard deviation of the
                    fragment library, you can probably assume that the
                    standard deviation is 10% of the average fragment
                    length.

--cap-mismatch-quals    Cap quality scores assigned at mismatch locations
                    to 2.  This was the default behavior in FLASH v1.2.7
                    and earlier.  Later versions will instead calculate
                    such scores as max(|q1 - q2|, 2); that is, the
                    absolute value of the difference in quality scores,
                    but at least 2.  Essentially, the new behavior
                    prevents a low quality base call that is likely a
                    sequencing error from significantly bringing down
                    the quality of a high quality, likely correct base
                    call.

-z, --compress          Compress the output files directly with zlib,
                    using the gzip container format.  Similar to
                    specifying --compress-prog=gzip and --suffix=gz,
                    but may be slightly faster.
```

```
--compress-prog=PROG    Pipe the output through the compression program
                      PROG, which will be called as 'PROG -c -',
                      plus any arguments specified by --compress-prog-args.
                      PROG must read uncompressed data from standard input
                      and write compressed data to standard output when
                      invoked as noted above.
                      Examples: gzip, bzip2, xz, pigz.

--compress-prog-args=ARGS
                      A string of additional arguments that will be passed
                      to the compression program if one is specified with
                      --compress-prog=PROG.  (The arguments '-c -' are
                      still passed in addition to explicitly specified
                      arguments.)
```

B Additional parameters for BWA

Additional parameters for FLASH can be accessed by execute command "bwa mem" in terminal. Here are all available additional options for bwa mapping:

```
        -k INT        minimum seed length [19]
        -w INT        band width for banded alignment [100]
        -d INT        off-diagonal X-dropoff [100]
        -r FLOAT      look for internal seeds inside a seed longer than {-k} * FLOAT [1.5]
        -y INT        seed occurrence for the 3rd round seeding [20]
        -c INT        skip seeds with more than INT occurrences [500]
        -D FLOAT      drop chains shorter than FLOAT fraction of the longest
                      overlapping chain [0.50]
        -W INT        discard a chain if seeded bases shorter than INT [0]
        -m INT        perform at most INT rounds of mate rescues for each read [50]
        -S            skip mate rescue
        -P            skip pairing; mate rescue performed unless -S also in use

Scoring options:

        -A INT        score for a sequence match, which scales options -TdBOELU
                      unless overridden [1]
        -B INT        penalty for a mismatch [4]
        -O INT[,INT]  gap open penalties for deletions and insertions [6,6]
        -E INT[,INT]  gap extension penalty; a gap of size k cost '{-O} + {-E}*k' [1,1]
        -L INT[,INT]  penalty for 5'- and 3'-end clipping [5,5]
        -U INT        penalty for an unpaired read pair [17]

        -x STR        read type. Setting -x changes multiple parameters unless overridden [n
                      pacbio: -k17 -W40 -r10 -A1 -B1 -O1 -E1 -L0  (PacBio reads to ref)
                      ont2d: -k14 -W20 -r10 -A1 -B1 -O1 -E1 -L0  (Oxford Nanopore 2D-reads t
                      intractg: -B9 -O16 -L5  (intra-species contigs to ref)

Input/output options:
```

```
-p            smart pairing (ignoring in2.fq)
-R STR        read group header line such as '@RG\tID:foo\tSM:bar' [null]
-H STR/FILE   insert STR to header if it starts with @; or insert lines in FILE [nul
-j            treat ALT contigs as part of the primary assembly (i.e. ignore
              <idxbase>.alt file)
-5            for split alignment, take the alignment with the smallest
              coordinate as primary
-q            don't modify mapQ of supplementary alignments
-K INT        process INT input bases in each batch regardless of
              nThreads (for reproducibility) []
-v INT        verbosity level: 1=error, 2=warning, 3=message, 4+=debugging [3]
-T INT        minimum score to output [30]
-h INT[,INT]  if there are <INT hits with score >80% of the max score,
              output all in XA [5,200]
-a            output all alignments for SE or unpaired PE
-C            append FASTA/FASTQ comment to SAM output
-V            output the reference FASTA header in the XR tag
-Y            use soft clipping for supplementary alignments
-M            mark shorter split hits as secondary
-I FLOAT[,FLOAT[,INT[,INT]]]
              specify the mean, standard deviation (10% of the mean if absent), max
              (4 sigma from the mean if absent) and min of the insert size distribut
              FR orientation only. [inferred]
```

C Additional parameters for VarScan2

Additional parameters for FLASH can be accessed by execute command "java -jar VarScan.jar" in terminal. Here are all available additional options for calling consensus and variants:

```
--min-coverage Minimum read depth at a position to make a call [8]
--min-reads2 Minimum supporting reads at a position to call variants [2]
--min-avg-qual Minimum base quality at a position to count a read [15]
--min-var-freq Minimum variant allele frequency threshold [0.01]
--min-freq-for-hom Minimum frequency to call homozygote [0.75]
--p-value Default p-value threshold for calling variants [99e-02]
--strand-filter Ignore variants with >90% support on one strand [1]
--variants Report only variant (SNP/indel) positions [0]
```