User Manual of VIcaller v1.1

January 24, 2019

Citation: Xun Chen, Jason Kost, Arvis Sulovari, Nathalie Wong, Winnie S. Liang, Jian Cao, and Dawei Li. A virome-wide clonal integration analysis platform for discovering cancer viral etiologies. *Genome Research*.

Download: www.uvm.edu/genomics/software/VIcaller

1 Introduction

Viral Integration caller (VIcaller) is a bioinformatics tool designed for identifying viral integration events using high-throughput sequencing (HTS) data. VIcaller is developed under Linux platform. It uses both FASTQ files or aligned BAM files as input. It also supports both single-end and paired-end reads. VIcaller contains one main Perl script, VIcaller.pl, that include three main functions: **1) detect**, which will detect virome-wide candidate viruses and integration events; **2) validate**, which will perform the *in silico* validation on those candidate viral integrations; **3) calculate**, which will calculate the integration allele fraction. We also generated a comprehensive viral reference genome library with 411,195 unique whole and partial genomes, covering all six virus taxonomic classes. The viral reference genome library also comes with a taxonomy database in a defined format that give virus name, and other information.

2 Availability

VIcaller is an open-source software. VIcaller.v1.1 source code is available at www.uvm.edu/genomics/software/VIcaller. It includes all Perl scripts, virome-wide reference library, and vector database.

3 VIcaller installation

3.1 Unzip the VIcaller installer

Unzip the installer and change the directory

```
$ tar vxzf VIcaller.tar.gz
$ cd VIcaller/
$ mkdir Tools
```

3.2 Install the dependent Perl libraries and tools

a) Currently VIcaller relies on the following dependencies to be compiled (contact Dr. Xun Chen if you need help get those tools or Perl libraries installed).
b) Obtain the installed file from the following links.
c) Follow the instruction to successfully install each tool (contact server manager if there is any compile issues).
d) Check or install the listed Perl libraries using cpan, cpanm or other methods.

Install each of the listed tools

- BWA (default version: v0.7.10): https://github.com/lh3/bwa/tree/master/bwakit
- Bowtie2 (default version: v2.2.7): https://sourceforge.net/projects/bowtie-bio/files/bowtie2/2.2.7/
- TopHat2 (v2.1.1): http://ccb.jhu.edu/software/tophat/index.shtml
- BLAT (default version: v.35): http://genomic-identity.wikidot.com/install-blat
- BLAST+ (default version: v2.2.30): http://mirrors.vbi.vt.edu/mirrors/ftp.ncbi.nih.gov/blast/executables/blast%2B/2.2.30/
- SAMtools (default version: v1.6): https://sourceforge.net/projects/samtools/
- HYDRA (default version: 0.5.3): https://code.google.com/archive/p/hydra-sv/downloads
- NGS QC Toolkit (default version: v2.3.3): http://genomic-identity.wikidot.com/install-blat

  a) Copy the script "TrimmingReads_sanger.pl" under the VIcaller/Scripts/ folder to the installed NGSQCToolkit_v2.3.3/Trimming/ folder

- FastUniq (Default version: v1.1): https://sourceforge.net/projects/fastuniq/
- SE-MEI (modified): https://github.com/dpryan79/SE-MEI (original version), the modified version can be found under the VIcaller/Scripts/ folder
  a) Copy the modified SE-MEI installer (SE-MEI-master.tar.gz) under the VIcaller/Scripts/ folder to the VIcaller/Tools/ folder
  b) Install the modified SE-MEI tool follow the README file
- RepeatMasker (default version: v4.0.5):
  a) Install RepeatMasker: http://www.repeatmasker.org/
  b) Install RMBlast aligner: http://www.repeatmasker.org/RMBlast.html
  c) Compile the Repbase database: https://www.girinst.org/repbase/
- MEME (default version: v4.11.1):
  http://web.mit.edu/meme_v4.11.4/share/doc/download.html
- TRF (default version: v4.07b): https://tandem.bu.edu/trf/trf.html

Install Perl libraries

```
$ cpan String::Approx
$ cpan Time::HiRes
$ cpan Test::Most
$ cpan Bio::Seq
$ cpan Bio::SeqIO
$ cpan Bio::DB::GenBank
$ cpan IO::Zlib
```

3.3 Prepare databases

Obtain and index the human reference genome using BWA, Bowtie2, and BLAST+ separately:

```
$ cd VIcaller/Database/Human/
$ wget http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.fa.gz
$ gunzip hg38.fa.gz
$ bwa index -a bwtsw hg38.fa
$ bowtie2-build hg38.fa hg38.fa
$ makeblastdb -in hg38.fa -dbtype nucl
```

Index the virome-wide library using BWA, Bowtie2, and BLAST+ separately:

```
$ cd VIcaller/Database/Virus/
$ bwa index -a bwtsw virus_db_090217.fa
$ bowtie2-build virus_db_090217.fa virus_db_090217.fa
$ makeblastdb -in virus_db_090217.fa -dbtype nucl
```

3.4 Prepare the VIcaller config file

3.4.1 Example of VIcaller.config

```
export PERL5LIB=/users/xchen/.cpan/build/
export PATH=$PATH:/users/xchen/VIcaller/Tools/bowtie2-2.2.7/
# human_genome = /users/xchen/VIcaller/Database/Human/hg38.fa
# human_genome_tophat = /users/xchen/VIcaller/Database/Human/hg38.fa
# virus_genome = /users/xchen/VIcaller/Database/Virus/virus_db_090217.fa
# virus_taxonomy = /users/xchen/VIcaller/Database/Virus/virus_db_090217.taxonomy
# virus_list = /users/xchen/VIcaller/Database/Virus/virus_db_090217.virus_list
# vector_db = /gpfs2/dli5lab/CAVirus/Database/Vector/Vector.fa
# cell_line = /users/xchen/VIcaller/Database/cell_line.list
# bowtie_d = /users/xchen/VIcaller/Tools/bowtie2-2.2.7/
# tophat_d = /users/xchen/VIcaller/Tools/tophat-2.1.1.Linux_x86_64/
# bwa_d = /users/xchen/VIcaller/Tools/bwa-master/
# samtools_d = /users/xchen/VIcaller/Tools/samtools-1.6/
# repeatmasker_d = /users/xchen/VIcaller/Tools/RepeatMasker/
# meme_d = /users/xchen/VIcaller/Tools/meme_4.11.1/
# NGSQCToolkit_d = /users/xchen/VIcaller/Tools/NGSQCToolkit_v2.3.3/
# fastuniq_d = /users/xchen/VIcaller/Tools/FastUniq/
# SE_MEI_d = /users/xchen/VIcaller/Tools/SE-MEI/
# hydra_d = /users/xchen/VIcaller/Tools/Hydra-Version-0.5.3/
# blat_d = /users/xchen/bin/x86_64/
# blastn_d = /users/xchen/VIcaller/Tools/ncbi-blast-2.2.30+-src/
```

3.4.2 Check the generated VIcaller.config file

#. Make sure the space between “#” and parameters.
#. Make sure the directory for the Perl library is correct or the libraries are available in the path if you install them locally.
#. Make sure the Bowtie2 directory is correct or it is available in the path (recommended) if you are going to analyze RNA-seq data.
#. Make sure the human and virus databases existed and correctly indexed.

4 VIcaller command line

```
$ perl VIcaller.pl <functions> [arguments]
```

4.1 Detect candidate viral integrations

4.1.1 Command line

```
$ perl VIcaller.pl detect [arguments]
```

4.1.2 Examples

a) WGS data in single-end fastq format:

```
$ perl VIcaller.pl detect -d WGS -i seq -f .fastq.gz -s single-end -t 12
```

b) RNA data in paired-end fastq format (set bowtie2 path before run the following command):

```
$ perl VIcaller.pl detect -d RNA-seq -i seq -f .fastq.gz -s paired-end -t 12
```

c) RNA alignment data in bam format (Note: Human reference genome should be the same as the bam file)

```
$ perl VIcaller.pl detect -d RNA-seq -i seq -f .bam -s paired-end -t 12
```

4.1.3 Parameters

-i|input_sampleID	sample ID (required)
-f|file_suffix	the suffix of the input data, including: .fq.gz|fastq.gz,.fq|fastq and .bam, indicate fastq and bam format separately default: .fq.gz (required)

-m|mode	running mode, including: standard, fast (default: standard)
-d|data_type	data type, including: WGS, RNA-seq (default: WGS)
-s|sequencing_type	type of sequencing data, including: paired-end, single-end (default: paired-end)
-t|threads	the number of threads will be used (default: 1)
-r|repeat	check repeat sequence
-a|align_back_to_human	reciprocal align back to the human reference genome
-q|QS_cutoff	quality score for each nucleotide
-c|config	user defined config file
-b|build	build version, including: hg19 and hg38 (default: hg38)
-h|help	print this help

4.2 Validate candidate viral integrations

4.2.1 Command line

```
$ perl VIcaller.pl validate [arguments]
```

4.2.2 Example

```
$ perl VIcaller.pl validate -i seq -S seq_1_24020575_24020787_HPV16_218931404 -G 218931404 -V HPV16
```

4.2.3 Parameters

-i|input_sampleID	sample ID (required)
-c|config	user defined configure file
-t|threads	the number of threads will be used (default: 1)
-S|String	string with sample ID, integration region, candidate virus, GI (required)
-G|GI	GI (required)
-V|Virus	candidate virus (required)
-h|help	print this help

4.3 Calculate allele fraction

4.3.1 Command line

```
$ perl VIcaller.pl calculate [arguments]
```

4.3.2 Example

```
$ perl VIcaller.pl calculate -i seq -f .fastq.gz -S -C 1 -P 24020575 -B 2 -N 20
```

4.3.3 Parameters

-i|input_sampleID	sample ID (required)
-c|config	user defined configure file
-t|threads	the number of threads will be used (default: 1)
-F|File_suffix_bam	the suffix of the input data, including: .fq.gz|fastq.gz,.fq|fastq and .bam, indicate fastq and bam format, default: .fq.gz (required)
-I|Index_sort	if the input file is sorted BAM format
-C|Chr	chromosome ID (required)
-P|Position	integration site (required)
-B|Breakpoint	both or one of upstream and downstream breakpoints detected, including: 1, 2 (default: 2)

```
-N|Number_reads         number of chimeric and split reads
-h|help                 print this help
```

5 Output

5.1 Output and file list

The candidate viral integrations detected by VIcaller are kept in the file with suffix of ".output" in Viral integration Format (VIF), with the visualization of the aligned read sequences in the file with suffix of ".visualization". After *in silico* validation and allele fraction calculation, the results are also kept in the output file. "seq" is an example sample ID.

Table 1 List of files produced by VIcaller

File name	Content
seq_h.sam	Alignment results in SAM format if the input is FASTQ file
seq_h1_h.sam	Secondary alignment in SAM format when the input is BAM file
seq_pe.bam	BAM file contained paired-end reads that both ends cannot be aligned to the human reference genome
seq_sm.bam	BAM file contained the end of chimeric reads that aligned to the human reference genome
seq_su.bam	BAM file contained the end of chimeric reads that not aligned to the human reference genome
seq_1.1fq	FASTQ file contained reads that only one end can be aligned to the human reference genome (forward)
seq_2.1fq	FASTQ file contained reads that only one end can be aligned to the human reference genome (reverse)
seq_1sf.fastq	FASTQ file contained soft-clipped sequences with ≥ 20 bp that were not aligned to the human reference genome
seq_1.1fuq	FASTQ file contained potential chimeric reads (forward)
seq_2.1fuq	FASTQ file contained potential chimeric reads (reverse)
seq_1sf.fuq	FASTQ file contained potential split reads
seq_1sf.othu	File contained soft-clipped sequences < 20 bp, that were aligned to the human reference genome
seq.type	File contained the read ID of all potential chimeric reads
seq.3	File contained records of both human and viral positions per read
seq.error	File contained records of both human and viral positions per read that were removed
seq_f2	File contained the visualization of chimeric and split reads of each candidate viral integration
seq_vsoft_sort.bam	BAM file contained the alignment results of the soft-clipped sequences against the viral reference genome library
seq_vsu.sort.bam	BAM file contained the alignment results of potential chimeric reads against the viral reference genome library

seq.virus_f	File contained the list of candidate viral integrations in VIF format
seq.virus_f2	File contained the list of high confident candidate viral integrations in VIF format
seq.visualization	File contained the visualization of chimeric and split reads of each high confident candidate viral integration
seq_1_24020575_24020787_human_papillomavirus_type_218931404.CS3	File contained *in silico* results for each chimeric and split reads
seq_1_24020701.allele_fraction	File contained the integration allele fraction for each candidate viral integration
seq.output	**Final output file containing the summary results of each candidate viral integration**

5.2 Header of the output file

Table 2 Header of the viral integration output file

Column	Header	Description
Col 1	Sample_ID	Sample ID
Col 2	VIcaller_mode	VIcaller running mode
Col 3	QC	If low quality nucleotide and reads were filtered
Col 4	Reciprocal_alignment	If the reads were reciprocal aligned back to the human reference genome
Col 5	Candidate_virus	Virus name
Col 6	GI	The selected, top one GenInfo Identifier (GI) for the integration
Col 7	Chr.	Human chromosome ID
Col 8	Start	Start position of the span genomic region of all chimeric and split reads in the human reference genome
Col 9	End	End position of the span genomic region of all chimeric and split reads in the human reference genome
Col 10	No._chimeric_reads	Total count of chimeric reads of the integration
Col 11	No._split_reads	Total count of split reads of the integration
Col 12	Upstream_breakpoint_on_human	Upstream breakpoint detected in the human reference genome
Col 13	Downstream_breakpoint_on_human	Downstream breakpoint detected in the human reference genome
Col 14	Upstream_breakpoint_on_virus	Upstream breakpoint detected in the viral genome
Col 15	Downstream_breakpoint_on_virus	Downstream breakpoint detected in theviral genome
Col 16	Information_of_both_upstream_and _downstream_breakpoints	Upstream and downstream breakpoint information. Upstream and downstream breakpoints were separated by semicolon; "D" and "E" represent if this breakpoint is detected by split reads (D), or estimated by chimeric reads separately (E); "+" and "-", represent the forward and reverse direction for both human (left) and virus (right) genome in the square per breakpoint; "na" represent this breakpint is not covered by any chimeric and split reads
Col 17	Integration_site_in_the_human_genome	Integration site in the human genome that was used for allele fraction detection. If both upstream and downstream breakpoints were detected, the medium position was used; If either one of the breakpoints were detected by split reads, this postion detected by split reads was used
Col 18	Integration_allele_fraction	Integration allele fraction value

Col 19	No._reads_supporting_nonVI	No. reads support no viral integration
Col 20	No._reads_supporting_VI	No. reads support viral integration, including chimeric and split reads
Col 21	Average alignment score	Average alignment score (AS) of reads support viral integration, including chimeric and split reads
Col 22	Is_cell_line_contamination	Is the integration from cell line contamination
Col 23	Is_vector	Is the integration from vector sequence
Col 24	Validation_chimeric_confident	*In silico* validation, the number of chimeric reads were consistently validated using BLASTN, BLAT and BWA-MEM
Col 25	Validation_chimeric_weak	*In silico* validation, the number of chimeric reads were validated by some but not all tools, including BLASTN, BLAT and BWA-MEM
Col 26	Validation_chimeric_false	*In silico* validation, the number of chimeric reads were false after validation
Col 27	Validation_split_confident	*In silico* validation, the number of split reads were consistently validated using BLASTN, BLAT and BWA-MEM
Col 28	Validation_split_weak	*In silico* validation, the number of split reads were validated by some but not all tools, including BLASTN, BLAT and BWA-MEM
Col 29	Validation_split_false	*In silico* validation, the number of split reads were false after validation

6 FAQ

6.1 Where can I get the human reference genome?

The hg38 reference genome can be download from this link: http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/. It is recommended to use the latest hg38.fa.gz file for indexing.

6.2 How to annotate the detected viral integrations?

The following Linux command can be used to extract the information required to run human genome functional annotation tools. The VIcaller output file is “seq.output”, and for example, if the functional annotation software is SnpEff, the following command line will extract the information required to run SnpEff. The output from using this command will be the input file for SnpEff.

```
$ awk '{if ($7!="Chr.")print$7"\t"$17"\t.\tA\tT\t."}' seq.output >SnpEff.intput
```

6.3 What is the difference between “Fast” mode and “Standard” mode?

“Fast” mode is significantly faster than “Standard” mode. However, the “Fast” mode does not analyze viral reads, which are supporting evidence for distinguishing between viral integrations and viral infections.

6.4 How to use the viral integration data from VIcaller for integration enrichment analysis?

VIcaller analyzes individual samples and then generates a list of viral integrations for each sample. Viral integration enrichment (bias) analysis, which is a statistical analysis, requires inclusion of a group of samples. The enrichment analysis has to be performed separately. There are multiple statistical models for calculating/determining enrichment hotspots (such as simulation-based Z score test). There are many available tools and R packages that can be selected for enrichment analysis. Users may have different preferences on statistical models to fit their actual samples/data.

6.5 Can I use the published tools that were designed for detecting transposable element insertions to identify virome-wide integrations?

VIcaller uses the reads that are commonly used in transposable element insertion and other structural variation detection tools. However, because VIcaller is specifically designed to identify virome-wide integrations, it has significant advantages for viral integration analysis over alignment-based transposable element insertion detection tools for viral integration analysis, which are designed to extract and mainly use (human’s) anomalous reads specifically. For example, 1) VIcaller supports the use of a virome-wide library as the reference to detect any characterized viruses, while most transposable element detection tools use transposable element sequences as the reference; and 2) VIcaller implements viral integration-specific quality control procedures and implements additional steps to *in silico* verify detected viral integrations. We have tried to compare VIcaller with other transposable element insertion detection software, e.g., MELT. MELT failed to run in a virome-wide fashion after we replaced MELT’s default consensus transposable element reference sequences with our virome-wide database. We further tested whether MELT was able to detect simulated candidate viral integrations, and we found that although MELT did run, it was not able to detect any of these integrations.