VarDict Overview

Version Number: 2.0b

|  |  |
| --- | --- |
| Document Details | |
| Document Title | VarDict Overview |
| Status / Version | Release/2.0b |
| Reference Code |  |

|  |
| --- |
| Disclaimer |
| If you are reading a copy of this document (i.e. print out of controlled electronic document or an electronic copy of a wet ink signed paper document) you may be reading an uncontrolled version. Before making updates, please ensure that you are using the latest version of this document and the associated template. |

|  |  |  |  |
| --- | --- | --- | --- |
| Document Approval / Review | | | |
| Criteria | Name & Role | Signature | Date |
| Author | Zaal Lyanov |  | 22-12-2016 |
| Approved |  |  |  |
| Approved |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Version Control | | | |
| Version | Date | Description of Changes | Author |
| 0.1 | 10/02/15 | Initial version | Dmitrii Adamian |
| 1.0 | 11/02/15 | Review and formatting | Liudmila Simonova |
| 1.1 | 20/02/15 | Added two more options into “Build Instruction” section | Viktor Kirst |
| 1.2 | 27/03/15 | Added a description of default one-thread mode | Viktor Kirst |
| 1.3 | 04/06/15 | Added command-line option ‘VS’ and other improvements | Zaal Lyanov |
| 1.4 | 24/06/15 | Improved local realignment | Zaal Lyanov |
| 1.4.5 | 16/03/16 | Bugfixes derived from Perl version | Zaal Lyanov |
| 1.4.7 | 06/12/16 | Bugfix, performance and resource usage improvements | Zaal Lyanov |
| 2.0b | 22/12/16 | Performance improvements and code refactoring | Zaal Lyanov |

|  |  |  |  |
| --- | --- | --- | --- |
| Referenced Documents | | | |
| No. | Document Title and ID | Version | Location |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1 Introduction 3](#_Toc411614749)

[2 Prerequisites 3](#_Toc411614750)

[3 Program Workflow 3](#_Toc411614751)

[4 Running Modes 3](#_Toc411614752)

[4.1 Single Sample Mode 3](#_Toc411614753)

[4.2 Paired Variant Calling 5](#_Toc411614754)

[4.3 Amplicon Based Calling 5](#_Toc411614755)

[5 Variant Finding Method 5](#_Toc411614756)

[5.1 CIGAR Preprocessing (Initial Realignment) 6](#_Toc411614757)

[5.2 Variants 6](#_Toc411614758)

[5.3 Variant Description String 6](#_Toc411614759)

[5.4 Variant Filtering 7](#_Toc411614760)

[6 Input Files 7](#_Toc411614761)

[6.1 BED File – Regions 7](#_Toc411614762)

[6.2 FASTA File - Reference Genome 8](#_Toc411614763)

[6.3 BAM File - Mapped Reads 8](#_Toc411614764)

# Introduction

VarDict is a variant discovery program written in Java and Perl. It can run in single sample, paired sample, or amplicon bias aware modes. As input, VarDict takes reference genomes in FASTA format, aligned reads in BAM format, and target regions in BED format.

# Prerequisites

Before running VarDict, make sure the following components are set up:

1. Java 1.8 or later
2. Perl
3. Samtools
4. R language

To see the help page for the program, run java –jar VarDict-0.1.jar -H.

# Program Workflow

The VarDict program follows the workflow:

1. Get regions of interest from a BED file or the command line.
2. For each segment:
   1. Find all variants for this segment in mapped reads:
      1. Optionally skip duplicated reads, low mapping-quality reads, and reads having a large number of mismatches.
      2. Skip a read if it does not overlap with the segment.
      3. Preprocess the CIGAR string for each read.
      4. For each position, create a variant. If a variant is already present, adjust its count using the adjCnt function.
   2. Realign some of the variants using special ad-hoc approaches.
   3. Calculate statistics for the variant, filter out some bad ones, if any.
   4. Assign a type to each variant.
   5. Output variants in an intermediate internal format (tabular).

**Note**: To perform Steps 1 and 2, use the Java program VarDict-0.1.

1. Perform a statistical test for strand bias using an R script.

**Note**: Use R script for this step.

1. Transform the intermediate tabular format to VCF. Output the variants with filtering and statistical data.

**Note**: Use the Perl scripts var2vcf\_valid.pl or var2vcf\_somatic.pl for this step.

# Running Modes

## Single Sample Mode

To run VarDict in single sample mode, use a BAM file specified without the “|“ symbol and perform Steps 3 and 4 (see the workflow in Section Error: Reference source not found) using teststrandbias.R and var2vcf\_valid.pl.

The following is an example command to run in single sample mode:

AF\_THR="0.01" # minimum allele frequency  
<path\_to\_vardict\_folder>/build/install/Vardict/bin/VarDict -G /path/to/hg19.fa -f $AF\_THR -N sample\_name -b /path/to/my.bam -z -F -c 1 -S 2 -E 3 -g 4 /path/to/my.bed | teststrandbias.R | var2vcf\_valid.pl -N sample\_name -E -f $AF\_THR

The following is an example command to run VarDict for a region (chromosome 7, position from 55270300 to 55270348, EGFR gene) with -R option:

<path\_to\_vardict\_folder>/build/install/Vardict/bin/VarDict -G /path/to/hg19.fa -f 0.001 -N sample\_name -b /path/to/sample.bam -z -R chr7:55270300-55270348:EGFR | teststrandbias.R | var2vcf\_valid.pl -N sample\_name -E -f 0.001 >vars.vcf

In single sample mode, output columns contain a description and statistical info for variants in the single sample. The following columns appear in the output file:

* ***Sample*** - sample name
* ***Gene*** - gene name from a BED file
* ***Chr*** - chromosome name
* ***Start*** - start position of the variation
* ***End*** - end position of the variation
* ***Ref*** - reference sequence
* ***Alt*** - variant sequence
* ***Depth*** - total coverage
* ***AltDepth*** - variant coverage
* ***RefFwdReads*** - reference forward strand coverage
* ***RefRevReads*** - reference reverse strand coverage
* ***AltFwdReads*** - variant forward strand coverage
* ***AltRevReads*** - variant reverse strand coverage
* ***Genotype*** - genotype description string
* ***AF*** - allele frequency
* ***Bias*** - strand bias flag
* ***PMean*** - mean position in read
* ***PStd*** - flag for read position standard deviation
* ***QMean*** - mean base quality
* ***Mapq*** - mean mapping quality
* ***QStd*** - flag for base quality standard deviation
* ***QRATIO*** - ratio of high quality reads to low-quality reads
* ***HIFREQ*** - variant frequency for high-quality reads
* ***EXTRAFR*** - Adjusted AF for indels due to local realignment
* ***SHIFT3*** - No. of bases to shifted to 3 prime for deletions due to alternative alignment
* ***MSI*** - MicroSattelite. > 1 indicates MSI
* ***MSINT*** - MicroSattelite unit length in bp
* ***NM*** - average number of mismatches for reads containing the variant
* ***HICNT*** - number of high-quality reads with the variant
* ***HICOV*** - position coverage by high quality reads
* ***5pFlankSeq*** - neighboring reference sequence to 5' end
* ***3pFlankSeq*** - neighboring reference sequence to 3' end
* ***SEGMENT:CHR\_START\_END*** - position description
* ***VARTYPE*** - variant type

## Paired Variant Calling

To run paired variant calling, use BAM files specified as BAM1|BAM2 and perform Steps 3 and 4 (see the workflow in Section Error: Reference source not found) using testsomatic.R and var2vcf\_somatic.pl.

In this mode, the number of statistics columns in the output is doubled: one set of columns is for the first sample, the other - for second sample.

The following is an example command to run in paired mode:

AF\_THR="0.01" # minimum allele frequency  
<path\_to\_vardict\_folder>/build/install/Vardict/bin/VarDict -G /path/to/hg19.fa -f $AF\_THR -N tumor\_sample\_name -b "/path/to/tumor.bam|/path/to/normal.bam" -z -F -c 1 -S 2 -E 3 -g 4 /path/to/my.bed | testsomatic.R | var2vcf\_somatic.pl -N "tumor\_sample\_name|normal\_sample\_name" -f $AF\_THR

## Amplicon Based Calling

Amplicon based calling mode is active if the BED file uses 8-column format and the -R option is not specified.

In this mode, only the first list of BAM files is used even if the files are specified as BAM1|BAM2 - like for paired variant calling.

For each segment, the BED file specifies the list of positions as start and end positions (columns 7 and 8 of the BED file). The Amplicon based calling mode outputs a record for every position between start and end that has any variant other than the reference one (all positions with the -p option). For any of these positions, VarDict in amplicon based calling mode outputs the following:

* Same columns as in the single sample mode for the most frequent variant
* Good variants for this position with the prefixes GOOD1, GOOD2, etc.
* Bad variants for this position with the prefixes BAD1, BAD2, etc.

The following columns appear in the output for good and bad variants: tcov, cov, rfc, rrc, fwd, rev, genotype, freq, bias, pmean, pstd, qual, qstd, mapq, qratio, hifreq, extrafreq.

For this running mode, the -a option (default: 10:0.95) specifies the criteria of discarding reads that are too far away from segments. A read is skipped if its start and end are more than 10 positions away from the segment ends and the overlap fraction between the read and the segment is less than 0.95.

# Variant Finding Method

VarDict processes aligned reads from a BAM file in the following order:

1. VarDict reads and processes all aligned reads in the BAM file sequentially. VarDict modifies the read alignment specified by the CIGAR string in some special cases (see the next sections). For each matched/unmatched base, insertion or deletion in the read, VarDict creates a variant and adds it to an intermediate data structure. If a variant is already present in this structure, VarDict updates its statistical characteristics.
2. VarDict performs special ad-hoc realignment of insertions, deletions, large insertions, and large deletions using unaligned parts of reads (soft-clipped ends). This step is optional and can be disabled using the -k 0 switch.
3. VarDict repacks all the variants into an intermediate data structure that holds the reference variant and all the other variants for each position in regions of interest.

## CIGAR Preprocessing (Initial Realignment)

Read alignment is specified in a BAM file as a CIGAR string (see Samtools documentation). VarDict modifies this string (and alignment) in the following special cases:

* Soft clipping next to insertion/deletion is replaced with longer soft-clipping. The same takes place if insertion/deletion is separated from soft clipping by no more than 10 matched bases.
* Short matched sequence and insertion/deletion at the beginning/end are replaced by soft-clipping.
* Two close deletions and insertions are combined into one deletion and one insertion
* Two close deletions are combined into one
* Two close insertions/deletions are combined into one
* Mis-clipping at the start/end are changed to matched sequences

## Variants

Simple variants (SNV, simple insertions, and deletions) are constructed in the following way:

* Single-nucleotide variation (SNV). VarDict inserts an SNV into the variants structure for every matched or mismatched base in the reads. If an SNV is already present in variants, VarDict adjusts its counts and statistics.
* Simple insertion variant. If read alignment shows an insertion at the position, VarDict inserts +BASES string into the variants structure. If the variant is already present, VarDict adjusts its count and statistics.
* Simple Deletion variant. If read alignment shows a deletion at the position, VarDict inserts -NUMBER into the variants structure. If the variant is already present, VarDict adjusts its count and statistics.

VarDict also handles complex variants (for example, an insertion that is close to SNV or to deletion) using specialized ad-hoc methods.

## Variant Description String

The description string encodes a variant for VarDict internal use.

The following table describes Variant description string encoding:

|  |  |
| --- | --- |
| **String** | **Description** |
| [ATGC] | for SNPs |
| +[ATGC]+ | for insertions |
| -[0-9]+ | for deletions |
| ...#[ATGC]+ | for insertion/deletion variants followed by a short matched sequence |
| ...^[ATGC]+ | something followed by an insertion |
| ...^[0-9]+ | something followed by a deletion |
| ...&amp;[ATGC]+ | for insertion/deletion variants followed by a matched sequence |

## Variant Filtering

A variant appears in the output if it satisfies the following criteria (in this order):

1. Frequency of the variant exceeds the threshold set by the -f option (default=5%).
2. The minimum number of high-quality reads supporting variant is larger than the threshold set by the -r option (default 2).
3. The mean position of the variant in reads is less than the value set by the -P option (default=5).
4. The mean base quality (phred score) for the variant is less than the threshold set by the -q option (default=23).
5. Variant frequency is more than 25% or reference allele does not have much better mapping quality than the variant.
6. Deletion variants are not located in the regions where the reference genome is missing.
7. The ratio of high-quality reads to low-quality reads is larger than the threshold specified by -o option (default=1.5).
8. Variant frequency exceeds 35%.
9. Mean mapping quality exceeds the threshold set by the -O option (default: no filtering)
10. In the case of an MSI region, the variant size is less than 8 nucleotides for the non-monomer MSI or 13 for the monomer MSI. Variant frequency is more than 20% for the non-monomer MSI and 27.5% for the monomer MSI.

# Input Files

## BED File – Regions

VarDict uses 2 types of BED files for specifying regions of interest: 4-column and 8-column. The 8-column file format is used for targeted DNA deep sequencing analysis (amplicon based calling), the 4-column file format - for single sample analysis.

All lines starting with #, browser, and track in a BED file are skipped. The column delimiter can be specified as the -d option (the default value is “:“).

The 8-column file format involves the following data:

* Chromosome name
* Region start position
* Region end position
* Gene name
* Score - not used by VarDict
* Strand - not used by VarDict
* Start position – VarDict starts outputting variants from this position
* End position –VarDict ends outputting variants from this position

The 4-column file format involves the following data:

* Chromosome name
* Region start position
* Region end position
* Gene name

## FASTA File - Reference Genome

The reference genome in FASTA format is read using samtools. For every invocation of the toVars function (usually 1 for a region in a BED file) and for every BAM file, a part of the reference genome is extracted from the FASTA file using the command:

samtools faidx $chr:$s\_start-$s\_end,

where

chr is the chromosome name

$s\_start is 700 bases before the start of region

$s\_end is 700 bases after the start of region.

## BAM File - Mapped Reads

Multiple BAM files can be specified with the “:” delimiter.

BAM files are converted to SAM format using samtools. For every invocation of the toVars function (usually 1 for a region in a BED file) a part of the BAM file is extracted with samtools. There is a possibility to apply the SAM filter for reads specified with command line arguments. VarDict runs samtools using the command:

samtools view BAM\_FILE\_NAME $chr:$START-$END,

where

$chr is the chromosome name

$START is the start of region

$END is the end of region.

VarDict redirects the output of this command to a pipe and reads it.