Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Sequence analysis

UMI-VarCal: a new UMI-based variant caller that efficiently improves low-frequency variant detection in paired-end sequencing NGS libraries

Vincent Sater [1,*], Pierre-Julien Viailly [2,3], Thierry Lecroq [1], Élise Prieur-Gaston [1], Élodie Bohers [2,3], Mathieu Viennot [2,3], Philippe Ruminy [2,3], Hélène Dauchel [2,3], Pierre Vera [1,2] and Fabrice Jardin [2,3]

[1] Normandie Univ, UNIROUEN, LITIS EA 4108, 76000 Rouen, France and
[2] Department of Pathology, Centre Henri Becquerel, Rouen, 76000, France and
[3] INSERM U1245, University of Normandie UNIROUEN, Rouen, 76000, France.

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Next Generation Sequencing (NGS) has become the go-to standard method for the detection of Single Nucleotide Variants (SNV) in tumor cells. The use of such technologies requires a PCR amplification step and a sequencing step, steps in which artifacts are introduced at very low frequencies. These artifacts are often confused with true low-frequency variants that can be found in tumor cells and cell-free DNA. The recent use of Unique Molecular Identifiers (UMI) in targeted sequencing protocols has offered a trustworthy approach to filter out artifactual variants and accurately call low frequency variants. However, the integration of UMI analysis in the variant calling process led to developing tools that are significantly slower and more memory consuming than raw-reads-based variant callers.
Results: We present UMI-VarCal, a UMI-based variant caller for targeted sequencing data with better sensitivity compared to other variant callers. Being developed with performance in mind, UMI-VarCal stands out from the crowd by being one of the few variant callers that don't rely on SAMtools to do their pileup. Instead, at its core runs an innovative homemade pileup algorithm specifically designed to treat the UMI tags in the reads. After the pileup, a Poisson statistical test is applied at every position to determine if the frequency of the variant is significantly higher than the background error noise. Finally, an analysis of UMI tags is performed, a strand bias and a homopolymer length filter are applied to achieve better accuracy. We illustrate the results obtained using UMI-VarCal through the sequencing of tumor samples and we show how UMI-VarCal is both faster and more sensitive than other publicly available solutions.
Availability: The entire pipeline is available at https://gitlab.com/vincent-sater/umi-varcal-master under MIT license.
Contact: vincent.sater@gmail.com

1 Introduction

Old traditional sequencing technologies have showed their limits and were rapidly replaced by next generation sequencing (NGS) for the detection of genomic aberrations like single nucleotide variants (SNV) and copy number variations (CNV). However, the use of such technologies requires extracted genomic DNA to be fragmented to produce DNA fragments. These fragments constitute the DNA library that has to be massively amplified in order to produce enough fragments and cover all the targeted regions. These fragments are finally sequenced by a NGS sequencer to generate reads. Nowadays, research centers rely heavily on next generation sequencers like Illumina or Thermo Fisher as their use produces very high coverage over targeted genomic regions, therefore allowing low-frequency variants to be accurately detected. In fact, the detection of low-frequency variants is a crucial step for cancer diagnosis. It is a very active area of research as it allows to personalize the treatment according to the found mutations. Unfortunately, low-frequency variants can be very easily confused with DNA polymerase errors produced during the amplification step as well as sequencing errors produced during the sequencing step. This has led to the rise of new sequencing protocols that rely on unique molecular identifiers (UMI) to correct the technical artifacts. The UMI implementation in such protocols was shown to be very effective in many published studies (Schmitt *et al.* (2012), Kukita *et al.* (2015), Newman *et al.* (2016), Young *et al.* (2016) and Bar *et al.* (2017)). UMIs are short arbitrary oligonucleotides sequences that are attached to the library of DNA fragments by ligation prior to the amplification step. The fact that UMIs are arbitrary sequences allows for every fragment to have a unique short oligonucleotide sequence attached to it, forming a unique tag for each fragment. These UMIs or unique tags are then amplified with their respective fragments and their sequences can be figured out from the reads through sequencing.

At the moment, three UMI-based variant callers are publicly available: DeepSNVMiner Andrews *et al.* (2016), MAGERI Shugay *et al.* (2017) and smCounter2 Xu *et al.* (2019). These tools all apply the same approach that tries to correct technical artifacts by performing a majority vote within a UMI family, since theoretically, reads that have the same UMI tag should be identical. By doing that, they build a consensus read for each UMI family and then they apply a statistical method (like Beta distribution) to model background error rates at each position and apply standard filters to call final variants. In order to call variants, raw-reads-based variant callers and UMI-based variant callers use SAMtools Li *et al.* (2009) to perform the pileup step. The pileup step generates a count of insertions, deletions and substitutions at each covered position in the BAM/SAM file. The advantage of SAMtools' pileup is that it is very efficient in terms of execution time and memory consumption. This allows for raw-reads-based variant callers to be relatively fast when compared to UMI-based variant callers. On the other hand, SAMtools does not take UMI tags into account so using it in a UMI-based variant caller significantly increases execution time. Only MAGERI does not use SAMtools pileup in its pipeline. By doing that, the tool is significantly slower and more memory consuming than all the other approaches, therefore justifying why other variant callers use it.

In this article, we present UMI-VarCal, a somatic single nucleotide variant and indel caller for UMI-based targeted paired-end sequencing protocols. UMI-VarCal stands out from the crowd by being one of the few variant callers that don't rely on SAMtools to do their pileup. Instead, thanks to an innovative homemade pileup algorithm specifically designed to treat the UMI tags present in the reads, UMI-VarCal is faster than both raw-reads-based and UMI-based variant callers. To test our tool, we compare it against two of the best raw-reads-based variant callers that only need the tumor sample to call variants, SiNVICT Kockan *et al.* (2017) and outLyzer Muller *et al.* (2016) and specifically designed to detect low-frequency variants. We also demonstrate that it can be as - if not more - sensitive as other UMI-based variant callers by comparing it against DeepSNVMiner.

2 Materials and methods

2.1 Samples

The Centre Henri Becquerel in Rouen designed a targeted sequencing panel for Diffuse Large B cell Lymphoma (DLBCL) analysis. This panel is designed to identify genomic abnormalities within a list of 36 genes that are most commonly impacted in this type of lymphoma. The panel was made specifically for QIAseq chemistry in order to introduce UMI during the construction of the library. For the list of genes used in the panel and the number of targeted regions per gene, the reader can refer to the supplementary table S1. In order to test UMI-VarCal against the three variant callers DeepSNVMiner, SiNVICT and outLyzer, we randomly selected 3 samples from a very large number of patients whose DNA were sequenced at the Centre Henri Becquerel and all suffering from DLBCL. Sample 1 and sample 3 are frozen biopsies extracted from 2 different patients at the Centre Henri Becquerel while sample 2 is a DNA extracted from a cell line. The selected samples have a corresponding histopathologic review and the quality of the DNA was checked to be adequate for sequencing.

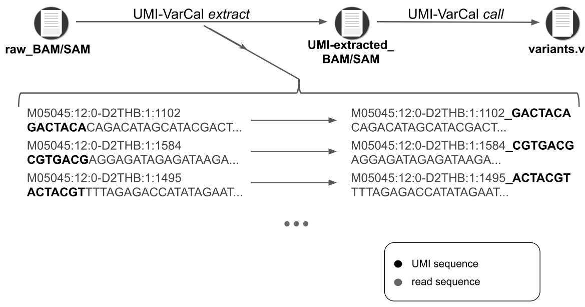


Fig. 1. Software input: UMI-VarCal can handle raw and UMI-extracted BAM or SAM files. If raw files are provided, the dedicated UMI extraction tool must be run prior to the calling tool. UMI tags are extracted from the read sequence and added to the end of the read ID. If BAM files are provided, they will be converted into SAM format. The variant calling tool can only start when the UMI-extracted SAM file is ready.

2.2 Software input

In order to run UMI-VarCal, three files are required: the paired-end SAM/BAM aligned file, the BED file containing the coordinates of the targeted genomic regions and a reference genome FASTA file with BWA index files. For ease of use, UMI-VarCal can accept BAM files as well as SAM files as input. We developed UMI-VarCal as a standalone variant caller that doesn't require any external tools to run. However, if the input is given under BAM format, SAMtools will be called in order to convert the BAM file into SAM format. Also, we integrated a UMI extraction tool in the software meaning that it can handle raw BAM/SAM files as well as UMI-extracted alignment files (Figure 1). Our tool can accept a fourth file under the PILEUP format. This file is only optional. In fact, when running UMI-VarCal on a sample, a PILEUP file is automatically produced. Giving this file to the software at execution will allow it to skip the pileup generation step (refer to section 2.3.1 for details) and load the old pileup instead. Being the step that takes most of the execution time, skipping it and loading the PILEUP file will make the user gain significant time.

2.3 Workflow

2.3.1 Pileup

The first step of the workflow is to generate the pileup. A pileup consists of the total count of match, substitution, insertion and deletion events at each position covered by the BED file. In fact, after filtering all the reads with low quality values, UMI-VarCal loops through every pair of reads and counts how many times each event was observed. Since each read is associated with a UMI tag, it means that every observed event at each position can be tagged with the corresponding UMI. After going through all the reads, this step will generate the complete list of A, C, G, T, insertion and deletion counts as well as their corresponding UMI tags at each position and for each chromosome. Also, this is when our algorithm estimates background error noise for each position. After completing the pileup, UMI-VarCal will automatically generate a PILEUP file with all the necessary informations. This file can be used if the user wishes to launch a new analysis of the same sample but with different variant calling parameters since changing these don't affect the pileup but only the variants called. In fact, this allows the analysis to complete faster since loading the pileup is very much faster that regenerating it.

2.3.2 Estimating the background error rate

In order to distinguish between real variants and technical artifacts (DNA polymerase and sequencing errors), the background error rate must be estimated. We already know that the background error rate is not constant and can vary at different positions so we can assume that each position has a specific error rate. In order to estimate site-specific error rates, some variant callers require a matched normal sample along with the tumor sample to make the analysis, while others use many control samples to model the error noise and provide the variant calling tool with the built-in model. While the first approach is definitely the best to estimate the error rate, matched normal samples are very hard to get, especially for cell-free DNA samples. Estimating the model on a number of control samples is a good approach that is capable of filtering many technical artifacts but has the limitation of being specific to the panel sequencing protocol and therefore, cannot be used across different panels. That's why UMI-VarCal uses base quality scores at each position to estimate the corresponding base error probabilities.

Since each sequenced base is associated with a quality score, we can use it to determine the base error probability for each position by calculating the average mean quality score. Assuming that X_i represents the total number of reads n covering the position i as $X_i = \{x_i^1, x_i^2, ..., x_i^n\}$, and that Q_i represents the quality scores of the bases at the position i for each read as $Q_i = \{q_i^1, q_i^2, ..., q_i^n\}$, we can easily calculate the average quality score of the position i by

$$\overline{q}_i = \frac{\sum\limits_{j=1}^{n} q_i^j}{n} \tag{1}$$

In fact, a quality score is a prediction of the probability of an error in base calling so the $\overline{q}_i$ that we calculated above reflects the sequencing quality or the base error probability at the position. Using the mean average qscore, we can compute the base error probability ϵ_i at each position i by

$$\epsilon_i = 10^{\frac{-\overline{q}_i}{10}} \tag{2}$$

2.3.3 Searching for candidate positions

The generated pileup contains the counts of A, C, G, T, insertions and deletions for all the positions covered in the BED file. UMI-VarCal loops through all these positions and applies at each one a Poisson test to determine if the alternative allele can be distinguished from the background error. We supposed that the presence of a variant is a rare event that could be treated as a hypothesis testing problem, where the null hypothesis (H_0) is that the alternative allele (substitutions, deletions and insertions) cannot be separated from background errors and the alternative hypothesis (H_1) is that the alternative allele can be distinguished from background errors and could actually represnt a true variant. At a position i, we define d_i as the depth at position i, ϵ_i as its base error probability and k_i as the total number of the alternative allele observations at position i. Under (H_0), k_i follows a Poisson distribution (λ_i) where λ_i is the number of errors expected to be found at a position i. We can simply calculate λ_i by

$$\lambda_i = d_i.\epsilon_i \tag{3}$$

At a position i, we can then calculate the p-value that represents the probability of observing more than λ_i errors as follows

$$p(k_i;\lambda_i) = 1 - \sum_{j=0}^{k_i} \frac{e^{-\lambda_i}.\lambda_i^j}{j!} \tag{4}$$

When we are conducting multiple hypothesis tests, we have an increased probability of false positives meaning that if we perform the same test multiple times, the chances of calling a null result as significant become higher. The false positive rate (FPR) refers to the number of false positives we expect when we perform a hypothesis test. So if we set the type 1 error probability (alpha) at 0.05, we can ensure that at worse, the percentage of false positives in all the tests we performed will be at 5%.. For example, if we test 10 000 positions and control the FPR at 0.05 (5%), on average 500 false variants (10000×0.05) will be called significant. This method poses a problem when we are conducting multiple tests as it becomes too permissive and we do not want to have such a great number of false positives. Typically, multiple comparison procedures control the false discovery rate (FDR) by trying to identify the most significant features and trying to filter out as much false positives as possible at the same time. UMI-VarCal applies the Benjamini-Hochberg procedure Benjamini and Hochberg (1995) in order to decrease the FDR, thus significantly reducing the number of total false positives. After applying the FDR correction to the p-values, we obtain the corresponding q-values. If the q-value is $\geq \alpha$, we can accept the null hypothesis and filter out the position as it means that the alternative allele observed at this position is most probably a technical artifact. Even with the FDR correction, the Poisson modeling applied to this situation maintains a relatively high sensitivity leaving us with a non negligible number of false positives. This is mainly due to the fact that the test does not take into consideration the strand bias nor the surrounding context of the variant. Therefore, in order to reduce the number of false positives, we apply three post-processing procedures as described below.

2.3.4 UMI analysis

When we apply the Poisson test to each position covered by the BED file, three scenarios are possible:

1. no alternative allele is found at the position: the position is filtered out
2. the q-value of the test is $\geq \alpha$ which means that the alternative allele is probably a technical artifact and therefore, the position is filtered out
3. the q-value of the test is < α which means that the alternative allele is most probably a true variant

In this last case only, a UMI analysis is applied. This step consists mainly of separating the list of all unique UMI tags found at a position into three different lists:

1. ref_umi: a list of all unique UMI tags found on the reads with the reference allele
2. alt_umi: a list of all unique UMI tags found on the reads with the alternative allele
3. $noise\_umi$: a list of all unique UMI tags found on the reads with neither the reference nor the alternative allele

Theoretically, if a variant is a true variant, it had to be found on the initial DNA fragment. So when we tag the DNA fragment with a unique UMI, we are also tagging the variant with the same unique UMI. After amplification, the DNA fragment is amplified and will produce thousands of reads, all

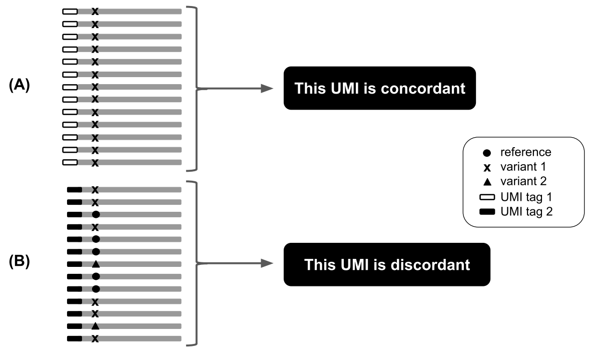


Fig. 2. The difference between a discordant UMI and a concordant UMI. (A) All the reads with the same green UMI tag present variant A: the green UMI tag is concordant. (B) The black UMI tag is found on 13 reads. Of the 13 reads, 6 present variant A, 5 present the reference allele and 2 present variant B. Since not all the reads have the same variant, we conclude that the black UMI tag is discordant.

carrying the same UMI as well as the same alternative allele. This means that at a specific position, if the alternative allele represents a true variant, all the reads that have the same UMI tag must present the same alternative allele. If that's the case, the UMI is called concordant (Figure 2A). On the other hand, if some of the reads with the same UMI tag present the reference or a noise allele, the UMI is called discordant (Figure 2B). Each concordant UMI tag characterizes a single DNA fragment. Using the three lists ref_umi, alt_umi and $noise\_umi$, we can calculate the number of concordant and discordant UMI tags for each variant. The more concordant UMI tags a variant has, the more DNA fragments it was present on initially. UMI-VarCal uses a concordant UMI tags threshold in order to filter out variants with too little concordant UMI counts. This UMI-based filter guarantees that the variants that pass through are not technical artifacts. These variants are called potential variants as they haven't passed through all the post-processing steps yet.

2.3.5 Strand bias filtering

This is the second filter and it is only applicable for potential variants (variants that have passed the Poisson test and the UMI analysis process). It was proven by Guo *et al.* (2012a) that a high strand bias (SB) could point out to a potential high false-positive rate, especially in Illumina short-read sequencing data. In this step, our strand bias filter calculates the strand bias score for each potential variant and, with the use of a threshold, aims to filter out all strand biased variants. Guo et al., 2012 compared three different methods to calculate the strand bias score (the traditional SB score, the GATK-SB score and the SB Fisher score) and demonstrated that the traditional SB calculation and the Fisher score can capture false positives better than the GATK-SB method. In addition, the traditional method used by Guo *et al.* (2012b) to detect false positives in variants from mitochondrial DNA samples showed very good results with a threshold of 1.0. UMI-VarCal uses the traditional SB calculation method (Equation 5) and applies the threshold of 1.0 in order to filter out the highest number of false positives among potential variants without being restrictive. We define R_f and R_r as the forward and reverse strands allele counts of the major allele, and V_f and V_r as the forward and reverse strands allele counts of the alternative allele.

$$SB = \frac{\mid \frac{V_f}{R_f+V_f} - \frac{V_r}{R_r+V_r} \mid}{\frac{V_f+V_r}{R_f+R_r+V_f+V_r}} \tag{5}$$

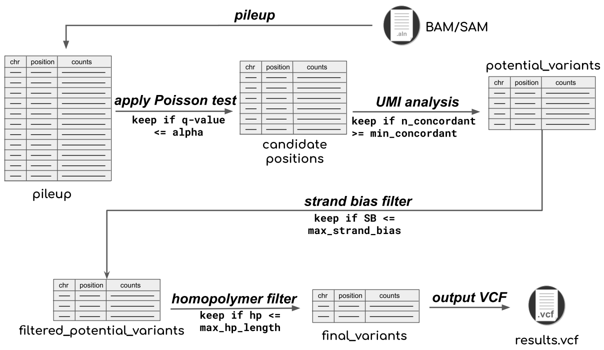


Fig. 3. UMI-VarCal workflow: UMI-Varcal starts by building a pileup from the BAM/SAM file provided. The pileup consists on A,C,G,T, insertion and deletion counts for each postilion covered by the alignment file. Since we know that not all positions contains mutations, UMI-VarCal starts looping through the pileup and applying a Poisson test at each position to filter out positions that don't - seem to - contain variants. To each of these candidate positions, an UMI analysis is carried out. At this step, one of 2 conditions are required in order to keep the variant. If the UMI analysis is successful, the potential variant must go through a strand bias filter to make sure that it isn't strand biased. If it passes that test, a final homopolymer region length filter is applied to make sure that the alternative allele is not due to the variant's presence in a long homopolymer region. If the alternative allele passes all these filters, it will be called and present in the final VCF file.

2.3.6 Filtering variants in homopolymer regions

Both pyrosequencing and ion semiconductor sequencing have difficulties to call correctly the bases situated in long homopolymer-containing regions. The uncertainty is due to the fact that the repeating identical nucleotides have to be incorporated during the same synthesis cycle. Ivády *et al.* (2018) demonstrated that the base calling accuracy suffers greatly as the length of the homopolymer region increases (> identical 4 bases). SomaticSniper Larson *et al.* (2012) is a variant calling tool that applies a homopolymer length filter in order to remove variants that occur in long homopolymer regions as this would mean that they are most probably artifacts due to sequencing errors. UMI-VarCal uses the same filter to remove variants found in a homopolymer region with a length > 7.

2.4 Implementation

The overall workflow (Figure 3) is comprised of Python modules that are called by a main Python script. All the modules are compiled in Cython to achieve better overall performance. UMI-VarCal is available for Python version 2 and 3. UMI-VarCal doesn't rely on any external program to launch. It only requires SAMtools if the input file is provided under BAM format.The extraction tool and the variant calling tool are executed through a UNIX command line interface. All the parameters and thresholds (minimum base quality, minimum read quality, minimum mapping quality, type 1 error rate alpha, minimum number of concordant UMI tags, maximum strand bias and maximum homopolymer region length) are customizable in order to allow the user total control over his results.

2.5 Software output

By default, UMI-VarCal automatically produces three files:

1. A standard VCF file containing all the variants that passed the tests and were successfully reported. For each variant, allele frequency, alternative allele observation count, total read depth, homopolymer length, variant type and confidence are provided. A confidence level is provided for each variant and is computed based on the variant's strand bias, homopolymer region length, q-value and the

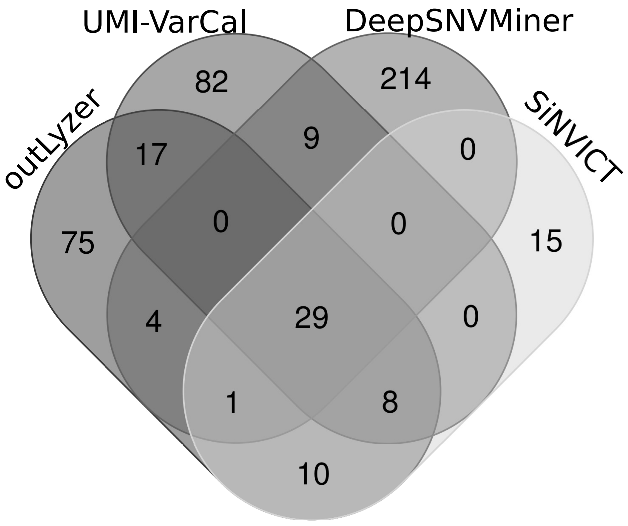


Fig. 4. Venn diagram of variants found by UMI-VarCal, DeepSNVMiner, SiNVICT and outLyzer in Sample 1

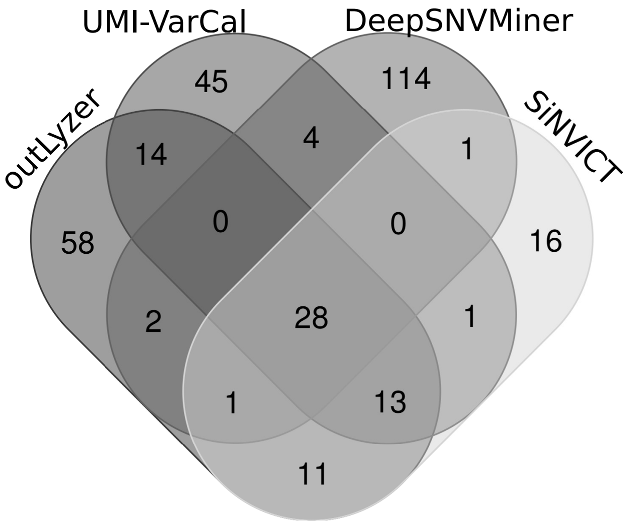


Fig. 5. Venn diagram of variants found by UMI-VarCal, DeepSNVMiner, SiNVICT and outLyzer in Sample 2

concordant/discordant ratio). Five levels are possible ranging from low to certain (low < average < high < strong < certain).

2. A VARIANTS file containing all the variants that were successfully reported. This file contains the same variants of the VCF file, in addition to detailed metrics for each variant.
3. A binary PILEUP file that corresponds to the entire pileup dumped. This file can be used to skip the pileup regeneration and load the pileup directly if the analysis was already done on the sample.

3 Results

At this moment, three UMI-based variant callers are publicly available: DeepSNVMiner, MAGERI and smCounter2. smCounter2 is relatively fast but has a theoretical detection limit of only 0.5%. MAGERI has a theoretical detection limit of 0.1% but is very slow and consumes a lot of memory (in our tests, it took 1 hour of execution time and a minimum of 200 GB of RAM to analyze one sample). Finally, DeepSNVMiner presents the advantage of having the same detection limit as MAGERI (0.1%) and is way more efficient in terms of execution time and memory consumption. To demonstrate our tool superiority over other available tools in terms of variant detection as well as performance, we compared it against DeepSNVMiner and also against two of the best raw-reads-based variant callers, outLyzer and SiNVICT that are designed specifically to detect low-frequency variants. In the following, we will compare the detection performance of the 4 variant callers on three different samples.

3.1 Variant detection comparison

3.1.1 Sample 1

In total, 464 variants were found (all the variants are detailed in Supplementary Table S2) (Figure 4). UMI-VarCal accounts for 145 variants, while DeepSNVMiner, outLyzer and SiNVICT detected 257, 144 and 63 variants respectively. Among these 145 variants, 29 are also found by all the other three tools and 63 were found by at least one other variant caller. 214 variants were only found by DeepSNVMiner: 139/214 didn't pass the Poisson test, 60/214 didn't pass the UMI analysis test, 4/214 are most probably strand biased and 1/214 is in a long homopolymer region. 75 variants were found only by outLyzer: 3/75 didn't pass the Poisson test, 64/75 didn't pass the UMI analysis test, 3/75 are most probably strand biased and 5/75 are in a long homopolymer region. 15 variants were found only by SiNVICT: 3/15 didn't pass the Poisson test, 7/15 didn't pass the UMI analysis test and 5/15 are detected in positions that are not covered by the provided BED file. 10 variants were found by both SiNVICT and outLyzer: all 10 variants are in a long homopolymer region. 4 variants were detected by both DeepSNVMiner and outLyzer: all four variants didn't pass the UMI analysis test and one of them is also most probably strand biased. 1 variant was found by DeepSNVMiner, SiNVICT and outLyzer: this variant is in a very long homopolymer region (length = 16). 82 variants were detected only by UMI-VarCal: 74/82 (90.2%) have a frequency below 1% and 28/82 (34.1%) have a frequency below 0.5%. UMI-VarCal detected 8 variants at a frequency < 0.4% but no variant was detected under the 0.3% frequency. Also, only 1/82 (1.2%) had a low level of confidence while 73/82 (89%) had at least a high confidence level.

3.1.2 Sample 2

In total, 308 variants were found (all the variants are detailed in Supplementary Table S3) (Figure 5). UMI-VarCal accounts for 105 variants, while DeepSNVMiner, outLyzer and SiNVICT detected 150, 127 and 71 variants respectively. Among these 105 variants, 28 are also found by all the other three tools and 60 were found by at least one other variant caller. 114 variants were only found by DeepSNVMiner: 63/114 didn't pass the Poisson test, 48/114 didn't pass the UMI analysis test and 3/114 are most probably strand biased. 58 variants were found only by outLyzer: 5/58 didn't pass the Poisson test, 46/58 didn't pass the UMI analysis test, 2/58 are most probably strand biased and 5/58 are in a long homopolymer region. 16 variants were found only by SiNVICT: 2/16 didn't pass the Poisson test, 7/16 didn't pass the UMI analysis test, 1/16 is in a long homopolymer region and 6/16 are detected in positions that are not covered by the provided BED file. 11 variants were found by both SiNVICT and outLyzer: 10/11 variants are in a long homopolymer region and one have 0 concordant UMI tags and therefore didn't pass the UMI analysis test. 2 variants were detected by both DeepSNVMiner and outLyzer: both of them didn't pass the UMI analysis test and one of them is also most probably strand biased. 1 variant was found by DeepSNVMiner, SiNVICT and outLyzer: this variant is in a long homopolymer region (length = 8) and has 0 concordant UMI tags. 45 variants were detected only by UMI-VarCal: 39/45 (86.7%) have a frequency below 1% and 13/45 (28.9%) have a frequency below 0.5%. UMI-VarCal detected 3 variants at a frequency

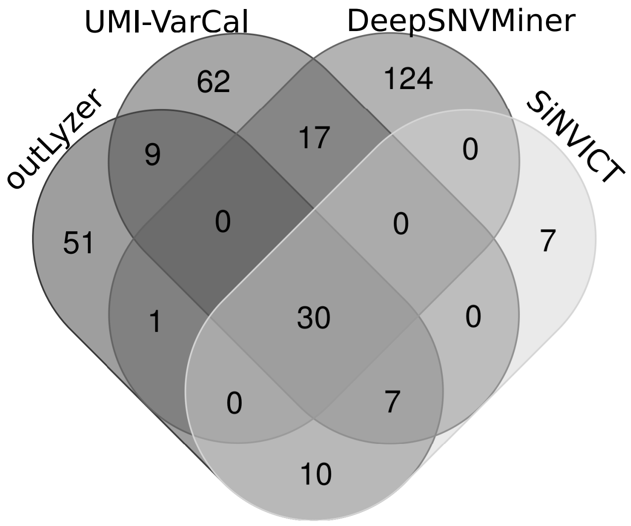


Fig. 6. Venn diagram of variants found by UMI-VarCal, DeepSNVMiner, SiNVICT and outLyzer in Sample 3

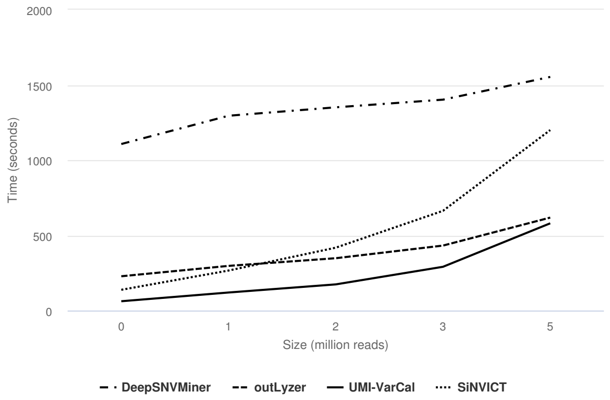


Fig. 7. Performance comparison between UMI-VarCal, DeepSNVMiner, SiNVICT and outLyzer

$<$ 0.4% but no variant was detected under the 0.3% frequency. Also, none of the 45 variants had a low level of confidence while 37/45 (82.2%) had at least a high confidence level.

3.1.3 Sample 3

In total, 318 variants were found (all the variants are detailed in Supplementary Table S4) (Figure 6). UMI-VarCal accounts for 125 variants, while DeepSNVMiner, outLyzer and SiNVICT detected 172, 108 and 54 variants respectively. Among these 145 variants, 30 are also found by all the other three tools and 83 were found by at least one other variant caller. 124 variants were only found by DeepSNVMiner: 88/124 didn't pass the Poisson test and 36/124 didn't pass the UMI analysis test. 51 variants were found only by outLyzer: 45/51 didn't pass the UMI analysis test, 1/51 are most probably strand biased and 5/51 are in a long homopolymer region. 7 variants were found only by SiNVICT: 3/7 didn't pass the UMI analysis test, 1/7 is in a long homopolymer region and 4/7 are detected in positions that are not covered by the provided BED file. 10 variants were found by both SiNVICT and outLyzer: 9/10 variants are in a long homopolymer region and one is most probably strand biased. 1 variant was detected by both DeepSNVMiner and outLyzer: this variant didn't pass the UMI analysis test. 62 variants were detected only by UMI-VarCal: 51/62 (82.3%) have a frequency below 1% and 10/62 (16.1%) have a frequency below 0.5%. UMI-VarCal detected 2 variants at a frequency $<$ 0.4% but no variant was detected under the 0.3% frequency. Also, none of the 62 variants had a low level of confidence while 55/62 (88.7%) had at least a high confidence level.

3.2 Performance comparison

In order to compare the performance of UMI-VarCal with the other three variant callers, we artificially created 5 different samples with increasing size (1, 2, 3, 5 and 10 million reads). This will allow to compare not only the performance of the tools but also to have a look at how the performance varies with sample size. All these tests are performed on a one core CPU running at 2.20 GHz. All measurements were done 3 times and the average was used for the comparison (Figure 7). To analyze 1 million reads, UMI-VarCal is the fastest with 62 seconds to complete the analysis. It is followed by SiNVICT that takes 138 seconds and outLyzer with 228 seconds. The slowest tool is DeepSNVMiner as it takes 1107 seconds to complete its analysis. For the 2 million reads analysis, the ranks don't change as UMI-VarCal is still the fastest of the four tools and DeepSNVMiner the slowest. The analysis of 3 million reads is the fastest on UMI-VarCal and the slowest on DeepSNVMiner as well. However, outLyzer sets the better time versus SiNVICT as the latter's performance seeming to suffer when increasing sample size. The ranks don't change at the 5 million reads mark as UMI-VarCal sill outperforms the three other variant callers and DeepSNVMiner being the slowest. Finally, the analysis of the 10-million-read sample is the fastest again on UMI-VarCal taking only 580 seconds to complete. outLyzer is closely behind and completes the analysis 38 seconds after (618 seconds). SiNVICT maintains the third place as it takes 1200 seconds to finish its analysis. DeepSNVMiner is still last with it taking 1553 seconds to have the final results ready. We note that both UMI-VarCal and outLyzer can be executed on multiple cores which can significantly decrease running time, especially on very large samples. In terms of memory consumption, UMI-VarCal is not very demanding. Memory consumption is not only impacted by the number of the reads but also by other factors such as the amplification factor of the sample and the maximum sequencing depth. Therefore, measuring the variation of memory consumtion with sample size is not significant. In our tests on the 3 DNA samples we selected, UMI-VarCal needed approximately 3 GB of RAM.

4 Discussion

Detecting somatic mutations with low allelic frequency is a challenge but is primordial in cancer studies in order to characterize tumor heterogeneity. Many raw-reads-based variant callers are available and do a good job in detecting most variants within a sample. Some of them however need a matched normal sample in order to perform the analysis: this can be problematic as these samples are difficult to find and might not exist in some applications. Other tools like SiNVICT and outLyzer do an outstanding job actually at detecting variants with frequencies as low as 0.5% but at the cost of having a high number of false positives: it is expected as these tools don't integrate a UMI analysis and thus cannot efficiently filter out false positives. UMI-based variant callers don't have this problem since they perform a UMI analysis that allows them to filter out most false positives. MAGERI showed some very good results in the publication with a theoretical detection limit of 0.1% but suffers in terms of performance as it consumes a lot of memory and is very slow. Another UMI-based variant caller is smCounter2 that has good performance but a detection limit of only 0.5%. DeepSNVMiner is a UMI-based variant caller that presents a theoretical detection limit of 0.1% and is relatively fast, compared to other

UMI-based variant callers. It starts by generating an initial list of variants using SAMtools calmd and then selects only those that have strong UMI support. However, in our tests, it seems to generate a lot of false positives since it doesn't contain a strand bias filter nor a homopolymer region length filter.

UMI-VarCal was able to perform better than the 3 other variant callers. It could easily detect the true variants found by the others and filter out the false positives due to its multi-step post-processing filters (UMI analysis filter, strand bias filter and homopolymer length filter). In addition, it was able to detect a high number of low-frequency variants (AF $\leq$ 1%) not found by other tools, of which 85% (on average) have at least a high level of confidence. In terms of execution time, we must admit that it is somewhat unfair to compare a UMI-based variant caller such as DeepSNVMiner to raw-reads-based variant callers. In our comparison, we showed not only that our tool can easily outperform an UMI-based variant caller but can only beat one of the fastest raw-reads based variant callers, outLyzer.

5 Conclusion

Here, we present UMI-VarCal: a standalone UMI-based variant caller developed to achieve more accurate low-frequency variant detection in paired-end sequencing NGS libraries. Also, thanks to a new pileup algorithm specifically designed to integrate the UMI tags in the reads, it is able to achieve excellent performance, in terms of both execution time and memory consumption, making it one of the fastest - if not the fastest - variant callers out there. In addition of its outstanding performance, UMI-VarCal is capable of detecting a large number of variants with frequencies as low as 0.3% that were completely missed out by the other tools. Among these variants, approximately 85% have at least a high confidence level meaning that they are most likely true variants. UMI-VarCal was built to allow total control for the user over his analysis since all the filters' parameters are customizable. This makes this tool adequate and available to a large number of clinical and research applications.

Funding

This work was partly funded by the University of Rouen Normandie and Vincent Sater is funded by a PhD fellowship from the Région Normandie.

References

Andrews, T. D., Jeelall, Y., Talaulikar, D., Goodnow, C. C., and Field, M. A. (2016). DeepSNVMiner: a sequence analysis tool to detect emergent, rare mutations in subsets of cell populations. *PeerJ*, **4**.

Bar, D. Z., Arlt, M. F., Brazier, J. F., Norris, W. E., Campbell, S. E., Chines, P., Larrieu, D., Jackson, S. P., Collins, F. S., Glover, T. W., and Gordon, L. B. (2017). A novel somatic mutation achieves partial rescue in a child with Hutchinson-Gilford progeria syndrome. *J Med Genet*, **54**(3), 212–216.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, **57**(1), 289–300.

Guo, Y., Li, J., Li, C.-I., Long, J., Samuels, D. C., and Shyr, Y. (2012a). The effect of strand bias in Illumina short-read sequencing data. *BMC Genomics*, **13**, 666.

Guo, Y., Cai, Q., Samuels, D. C., Ye, F., Long, J., Li, C.-I., Winther, J. F., Tawn, E. J., Stovall, M., Lähteenmäki, P., Malia, N., Levy, S., Shaffer, C., Shyr, Y., Shu, X.-o., and Boice, J. D. (2012b). The use of Next Generation Sequencing Technology to Study the Effect of Radiation Therapy on Mitochondrial DNA Mutation. *Mutat Res*, **744**(2), 154–160.

Ivády, G., Madar, L., Dzsudzsák, E., Koczok, K., Kappelmayer, J., Krulisova, V., Macek, M., Horváth, A., and Balogh, I. (2018). Analytical parameters and validation of homopolymer detection in a pyrosequencing-based next generation sequencing system. *BMC Genomics*, **19**.

Kockan, C., Hach, F., Sarrafi, I., Bell, R. H., McConeghy, B., Beja, K., Haegert, A., Wyatt, A. W., Volik, S. V., Chi, K. N., Collins, C. C., and Sahinalp, S. C. (2017). SiNVICT: ultra-sensitive detection of single nucleotide variants and indels in circulating tumour DNA. *Bioinformatics*, **33**(1), 26–34.

Kukita, Y., Matoba, R., Uchida, J., Hamakawa, T., Doki, Y., Imamura, F., and Kato, K. (2015). High-fidelity target sequencing of individual molecules identified using barcode sequences: de novo detection and absolute quantitation of mutations in plasma cell-free DNA from cancer patients. *DNA Res*, **22**(4), 269–277.

Larson, D. E., Harris, C. C., Chen, K., Koboldt, D. C., Abbott, T. E., Dooling, D. J., Ley, T. J., Mardis, E. R., Wilson, R. K., and Ding, L. (2012). SomaticSniper: identification of somatic point mutations in whole genome sequencing data. *Bioinformatics*, **28**(3), 311–317.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**(16), 2078–2079.

Muller, E., Goardon, N., Brault, B., Rousselin, A., Paimparay, G., Legros, A., Fouillet, R., Bruet, O., Tranchant, A., Domin, F., San, C., Quesnelle, C., Frebourg, T., Ricou, A., Krieger, S., Vaur, D., and Castera, L. (2016). OutLyzer: software for extracting low-allele-frequency tumor mutations from sequencing background noise in clinical practice. *Oncotarget*, **7**(48), 79485–79493.

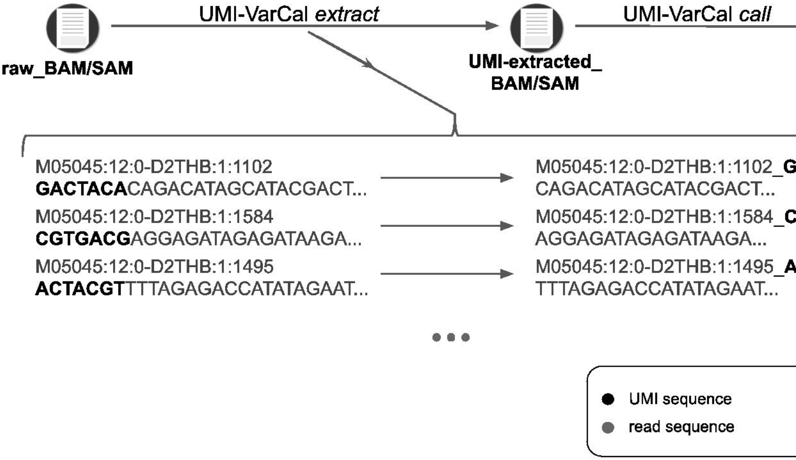
Newman, A. M., Lovejoy, A. F., Klass, D. M., Kurtz, D. M., Chabon, J. J., Scherer, F., Stehr, H., Liu, C. L., Bratman, S. V., Say, C., Zhou, L., Carter, J. N., West, R. B., Sledge, G. W., Shrager, J. B., Loo, B. W., Neal, J. W., Wakelee, H. A., Diehn, M., and Alizadeh, A. A. (2016). Integrated digital error suppression for improved detection of circulating tumor DNA. *Nat Biotechnol*, **34**(5), 547–555.

Schmitt, M. W., Kennedy, S. R., Salk, J. J., Fox, E. J., Hiatt, J. B., and Loeb, L. A. (2012). Detection of ultra-rare mutations by next-generation sequencing. *Proc Natl Acad Sci U S A*, **109**(36), 14508–14513.

Shugay, M., Zaretsky, A. R., Shagin, D. A., Shagina, I. A., Volchenkov, I. A., Shelenkov, A. A., Lebedin, M. Y., Bagaev, D. V., Lukyanov, S., and Chudakov, D. M. (2017). MAGERI: Computational pipeline for molecular-barcoded targeted resequencing. *PLoS Comput Biol*, **13**(5).

Xu, C., Gu, X., Padmanabhan, R., Wu, Z., Peng, Q., DiCarlo, J., and Wang, Y. (2019). smCounter2: an accurate low-frequency variant caller for targeted sequencing data with unique molecular identifiers. *Bioinformatics*, **35**(8), 1299–1309.

Young, A. L., Challen, G. A., Birmann, B. M., and Druley, T. E. (2016). Clonal haematopoiesis harbouring AML-associated mutations is ubiquitous in healthy adults. *Nat Commun*, **7**.



M05045:12:0-D2THB:1:1102
GACTACACAGACATAGCATACGACT...

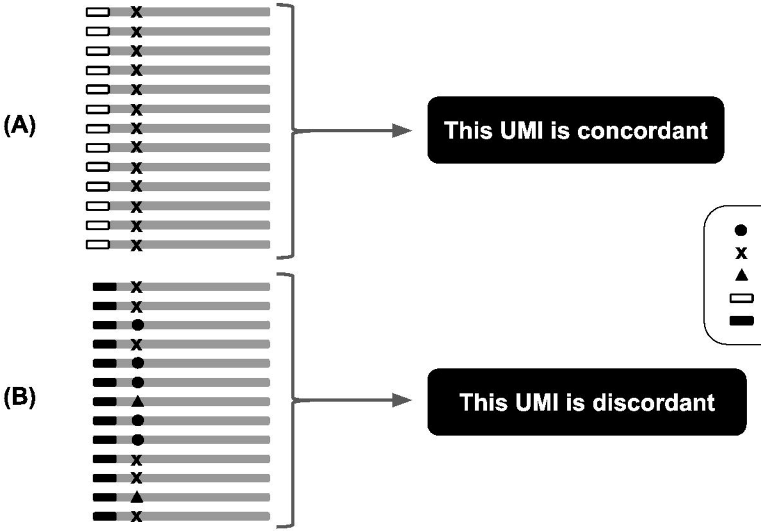
M05045:12:0-D2THB:1:1584
CGTGACGAGGAGATAGAGATAAGA...

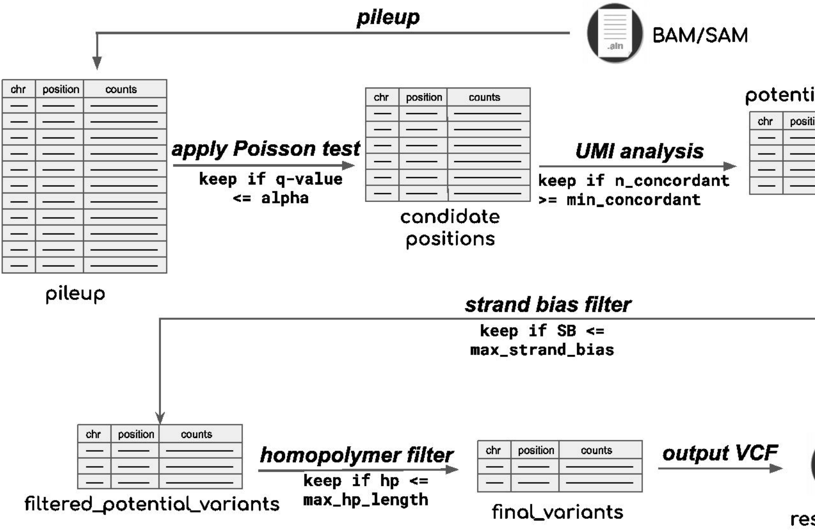
M05045:12:0-D2THB:1:1495
ACTACGTTTTAGAGACCATATAGAAT...

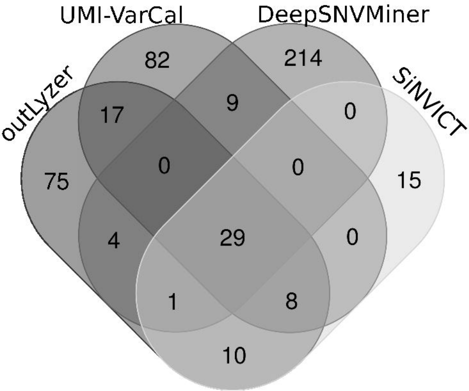
M05045:12:0-D2THB:1:1102_**G**
CAGACATAGCATACGACT...

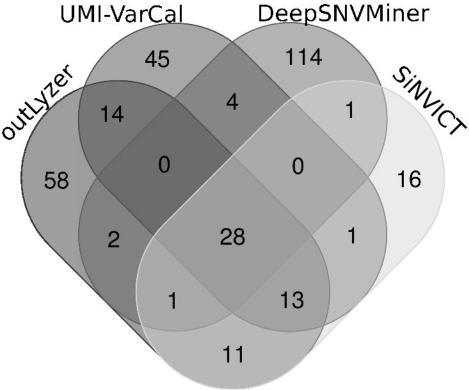
M05045:12:0-D2THB:1:1584_**C**
AGGAGATAGAGATAAGA...

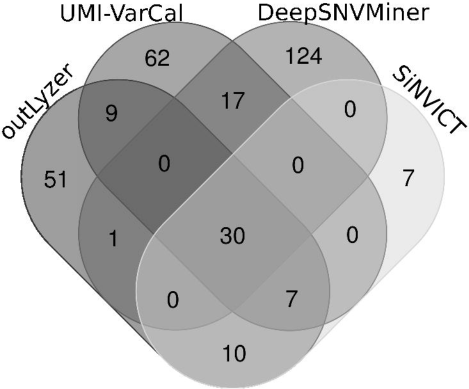
M05045:12:0-D2THB:1:1495_**A**
TTTAGAGACCATATAGAAT...

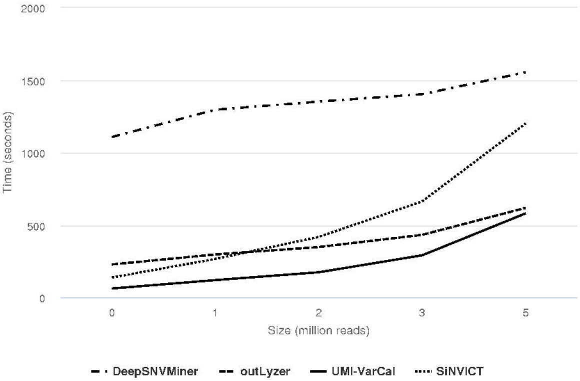

(A)
This UMI is concordant
(B)
This UMI is discordant


pileup
.aln
BAM/SAM
chr
position
counts
pileup
apply Poisson test
keep if q-value <= alpha
candidate positions
UMI analysis
keep if n_concordant >= min_concordant
potenti
positi
strand bias filter
keep if SB <= max_strand_bias
filtered_potential_variants
homopolymer filter
keep if hp <= max_hp_length
final_variants
output VCF
res


UMI-VarCal
DeepSNVMiner
outLyzer
SiNVICT
82
214
17
9
0
75
0
0
15
4
29
0
1
8
10


UMI-VarCal
DeepSNVMiner
outLyzer
SiNVICT
45
114
14
4
1
58
0
0
16
2
28
1
1
13
11


UMI-VarCal
DeepSNVMiner
outLyzer
SiNVICT
62
124
9
17
0
51
0
0
7
1
30
0
0
7
10


2000
1500
1000
500
0
Time (seconds)
0
1
2
3
5
Size (million reads)
DeepSNVMiner
outLyzer
UMI-VarCal
SiNVICT