Improved VCF Normalization for Evaluation of DNA Variant Calling Algorithms

Arash Bayat Bruno Gaeta
Aleksandar Ignjatovic Sri Parameswaran

University of New South Wales, Australia
{a.bayat, bgaeta}@unsw.edu.au, {ignjat, sridevan}@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-201601
Feb 2016

THE UNIVERSITY OF
NEW SOUTH WALES

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

Variant Call Format (VCF) is widely used to store data about genetic variations. Applications include evaluation of variant calling workflows and the study of the similarity of individual variations, where it is required to compare two sets of variants against each other. However, finding concordance between VCF files is a complicated task as the same variant can be represented in several different ways and is therefore not necessarily reported in a unique way by different software. In this paper, we have introduced a VCF normalization method that results in more accurate comparison. Basically, in our proposed normalization procedure, we apply all variations in a VCF file to the reference genome to created a mutated genome, and then recall variants by aligning this mutated genome back with the reference genome. The normalized VCF is not necessarily closer to the truth but is suitable for comparison purposes. The result shows over 34 times less disagreement when comparing VCF files normalized by our method relative to unormalized files. Our method mostly relies on available validated software.

1 Background

Variant calling workflows are evaluated by comparing their discovered variants (call set) with a set of truth variants (true set) that can be simulated variants or set of variants identified by a more reliable technique. In this comparison, we specifically look for the number of variants listed in call set that exist in the true set (True Positives: TP), those reported but not observed in the true set (False Positives: FP), and variants missed by the variant caller (False Negatives: FN). Several evaluation studies [1] [2] [3] have been performed, including [3] Xiangtao et al. who utilized *DWGsim*[1][5] to simulate human genome variations, and the *VCFtools*[6] compare module to compare VCF files.

As we know, *VCFtools* does not consider different representations of variants during the comparison. Hence, Adrian et al. in [7] mathematically prove that each variant could be represented in a normalized form. Their proposed variant normalization algorithm is implemented in a toolkit called *vt normalize*. An alternative normalization method is *GATK* [8] [9] *LeftAlignAndTrimVariant*, however, Adrian shows that their method is more efficient in normalizing variants. Unfortunately, normalizing individual variant does not consider equivalency of sets of variants (referred as sets in the rest of this text). To illustrate the complexity of recognizing identical sets, Figure 1.1 represents a tiny reference sequence and three sets each including three variants. All three sets are equivalent to each other as they result in same mutated sequence, however none of the variants are recognized as similar by *VCFtools*. Since *vt normalize* successfully normalizes the last variant (insertion of C), it is determined as a true positive by *VCFtools* comparing normalized sets. However, the uniqueness of the first two variants remains undiscovered.

Pos: 12345678901234567890123456 7 8901
Ref: TGCATGATGCAC**T**CC**G**TTGCATCCCCC-TGAG
Mut: TGCATGATGCAC-CC-TTGCATCCCCC**C**TGAG

	POS	REF	ALT	Description
VCF1:	11	ACT	A	Delete CT
	16	G	C	SNP G -> C
	23	C	CC	Insert C
VCF2:	12	CT	C	Delete T
	15	CG	C	Delete G
	24	C	CC	Insert C
VCF3:	13	T	C	SNP T -> C
	14	CCG	C	Delete CG
	25	C	CC	Insert C

Figure 1.1: Example of three equivalent variant sets. Pos: Right most digit of position in the sequence. Ref: Reference sequence. Mut: Mutated sequence. VCF1, VCF2, VCF3: three sets of variants all transform Ref to Mut. Ref and Mut are aligned together.

Another VCF comparator software is the *Next generation sequencing Error Analysis Toolkit (NEAT)*[10] compare module. Although its algorithm has not published yet, it identifies three perfect matches when comparing the sets in Figure 1.1. Nevertheless, the example in Figure 1.2 reveals *NEAT's* weakness.

[1] *DWGsim* is based on *wgsim* bundled in *SAMtools*[4]

In this example, the resulting mutated sequences differ in a single base (a symbol in a sequence) while *NEAT* reports two FPs and two FNs. In this case even normalizing sets with *vt normalize* could not change the outcome.

Pos :1234567890 123				**Pos** :1234567890123
Ref :AAACG**TG**TAT--AAA				**Ref** :AAACG**T**G**T**A**T**AAA
Mut1:AAACG--TAT**AA**AAA				**Mut2**:AAACG**C**TA**A**AAA
	POS	**REF**	**ALT**	**Description**
VCF1:	5	GTG	G	Delete TG
	10	T	TAA	Insert AA
VCF2:	7	G	C	SNP G -> C
	10	T	A	SNP T -> A
Mut1:	AAACGT**A**TAAAAA			
Mut2:	AAACGT**C**TAAAAA			

Figure 1.2: Example of issue with *NEAT*. Pos: Right most digit of position in the sequence. Ref: Reference sequence. VCF1, VCF2: Variant sets. Mut1, Mut2, Mutated Sequence by VCF1 and VCF2 respectively.

2 Our Method

To mitigate issues discussed in Section 1 we propose a VCF normalization method which takes into account all the variants presented in VCF file instead of focusing on a single variant. Our method can be abstracted into two sub-processes: injecting all mutations from the input VCF file into the reference sequence to produce the mutated sequence, then recalling all variants by aligning both sequences together. Since equivalent sets will result in exactly the same mutant sequence, independent from the representation issue, the former sub-process leads to a sole output. Furthermore, due the singularity of the rediscovery approach in latter sub-processes, the recalled set is similar if the given sequences are the same.

Although the above procedure simply supports homozygous variants [1], to handle heterozygous variants [2] we use a trick. We divide the input VCF file into two where each heterozygous variant appears only in one of the output files. For this purpose, we suggest to use the phasing [3] if available; Otherwise, random phasing works as well. There might be rare cases, discussed in Section 5, where random phasing might cause inaccuracy. The output VCF files are then normalized independently, and merged to produce the final output.

Our proposed method does not normalize individual variants but the whole VCF file. Thus, we strongly recommend, as it is necessary, to use *vt normalize* to process our produced output file. We also advise the use of *vt normalize* to process the input VCF file as some VCF files contain errors which are detected and sometimes fixed by *vt normalize*. Details are discussed in Section 5.

Figure 2.1 elaborates on the normalization procedure in details. In the beginning, the input VCF is processed using *vt normalize*. Then, the scripts we

[1] Variants appear in both copy of a chromosome uniquely.

[2] Variants appear differently in each copy of a chromosome.

[3] A data field in the VCF file that describes how each chromosome in a chromosome pair is affected by a heterozygous variant.

wrote divide it into two based on the genotype of the variants. *GATK FastaAlternateReferenceMaker* (FARM) produces a mutated genome from the reference and the given VCF file. *NUCmer*, *delta-filter*, and *show-snps* from the *MUMmer* [11] [12] [13] whole genome alignment toolkit are used to align the genomes, filter the best alignment, and recall variations respectively. A modified version of a script, taken from the web[14], converts the *MUMmer* snps file into VCF files. Finally the normalized VCF files are combined using *GATK CombineVariants*, and normalized using *vt normalize*. It is important to note that our approach does not consider the quality of variants, and it is purely quantitative.

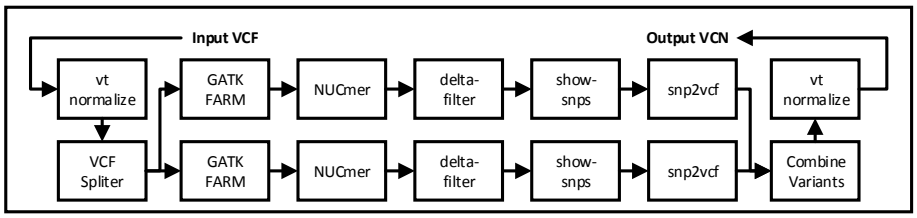

Figure 2.1: Proposed Normalization Scheme

3 Experimental Setup and Evaluation Method

As a case study, we simulate individual variants and short reads from chromosome 20 of the human genome using DWGsim with the following options. All other options remain default.

- Variation ratio is set to 0.002.
- Paired-end short read length is set to 100.
- Folded coverage is set to 20.

Short reads are mapped using *BWA MEM* [15] and sorted using *Picard* [16]. Then variants are called from mapped reads using *SAMtools* [4], *SNVer* [17], *GATK UnifiedGenotyper* and *GATK HaplotypeCaller*. The simulated variants generated by *DWGsim* and those called by the above variant callers are then normalized by both *vt normalize* alone (referred as VTN files) and the procedure represented in Figure 2.1 (referred to as VCN files).

We use *VCFtools*, *NEAT*, and *NEATf* (NEAT fast mode with no equivalency check) VCF comparators to find the concordance (TP, FP, and FN) between each discovered VCF/VTN/VCN file against the simulated VCF/VTN/VCN file. Since each of the comparison algorithms considers different criteria to check for equivalency of variants, in most cases, their outcome slightly differ while comparing the same input. *Picard* and *GATK* also include a VCF compare modules, however due to incompatibility with other variant callers' output, we decided to not to use them.

Since no mathematically proven definition of best-normalized variants set has been proposed yet, we define a better normalization technique as one which results in a higher level of agreement between the outcome of various compare algorithms. If the difference between the number of TP, FP, and FN reported

by different comparators is lower, this means there are less left for comparison algorithm to optimize and better normalization has been applied. Note that this definition is entirely independent of the accuracy of the variant caller. The reason that several variant callers are used is to show how different VCF files, produced by different algorithms, can be normalized.

Based on the definition above, we determine our evaluation metric in the following manner. First, for each TP, FP, and FN of each comparison, we sum up the differences and name it Disagreement Factor (DF). For instance, if 10, 15, 13 are reported as the number of FNs by different comparators, then DF is equal to [(15-10)+(13-10)+(15-13)]=10. We sum up DF for all comparison on the same normalization level (VCF/VTN/VCN) and name it Total Disagreement Factor (TDF). We expect VCN results in the lowest TDF and VCF (un-normalized files) results in the highest TDF.

Finally to check that our normalized VCF file, VCN file, is equivalent to the original VCF file, we used *GATK FARM* to create mutated genome sequences from both VCF and VCN files. Both genome sequences were compared against each other using the *Linux diff* which showed they were identical.

4 Results

Table 4.1 summarizes all comparison results and Figure 4.1 shows the TDF for each metric (TP, FP, and FN) separately as well as summed together on a log scale. The sum of TDF for VCF is over 34 times larger than that for VCN while the difference between VCF and VTN is small.

Table 4.1: Comparison Result

Variant Caller	Compare Algorithm	Un-normalized VCF			VTN Normalized			VCN Normalized		
		TP	FN	FN	TP	FN	FN	TP	FN	FN
SNVer	VCFtools	86157	32718	13	86160	32715	10	86159	32730	10
	NEATf	86153	32726	27	86153	32721	27	86161	32724	19
	NEAT	86157	32722	24	86157	32717	24	86161	32724	19
SAMtools	VCFtools	104984	8347	4586	109756	8747	4986	105263	13319	3525
	NEATf	105199	13680	9918	109832	9042	5285	105110	13775	3992
	NEAT	109264	9615	5854	109840	9034	5278	105114	13771	3988
GATK Unified Genotyper	VCFtools	109818	9046	6278	111788	7076	4308	111796	7086	4297
	NEATf	106114	12765	5605	107891	10983	3828	111616	7269	4472
	NEAT	107870	11009	3849	107899	10975	3820	111624	7261	4465
GATK Haplotype Caller	VCFtools	38589	80295	919	38697	80187	811	38699	80200	809
	NEATf	23050	95829	264	23108	95766	206	38686	80199	823
	NEAT	23108	95771	206	23110	95764	204	38686	80199	823

5 Limitation

Although our normalization method works well when a realistic variation ratio is considered there are rare cases in which our scheme does not fit well. In our normalization scheme, we recall variants base on the best alignment. When comparing two highly discordant VCF files with a high concentration of variations, specially indels, in a small region of the genome, it is possible for *NUCmer* to produce different alignments for each mutated genome on such regions. In spite of the fact that the recalled variants are still valid, they are not well prepared for the purpose of comparison and may result in a higher number of FPs and FNs than expected.

Another limitation exists when the phasing is not available in an input VCF file containing complex multi-allelic variants. There could be extremely rare

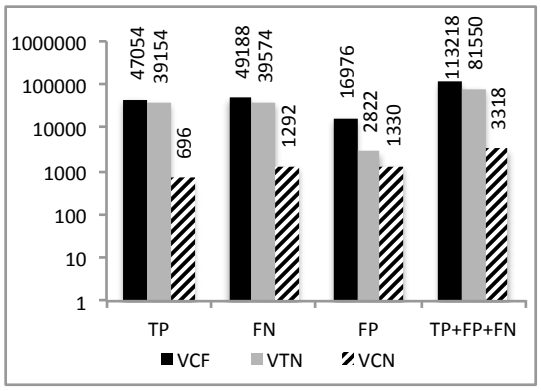

Figure 4.1: TDF for TP, FP, FN and sum of them.

cases where considering random phasing results in different normalized VCF. Figure 5.1 is a hypothetical example for such case.

Pos:	1234 56 789		**Pos:**	1234 56 789	
A1x:	ACT**A**--CC**CC**TCG		**A2x:**	ACT**A**--CC--TCG	
Ref:	ACT**G**--CC--TCG		**Ref:**	ACT**G**--CC--TCG	
A1y:	ACT**GCC**CC--TCG		**A2y:**	ACT**GCC**CC**CC**TCG	
	POS	**REF**	**ALT**	**Phase1**	**Phase2**
VCF	4	G	A,GCC	x|y	x|y
IN:	6	C	CCC,C	x|y	y|x
Ref:	ACT**G**CC--TCG		**Ref:**	ACT**G**CC----TCG	
A1x:	ACT**A**CCCCTCG		**A2x:**	ACT**A**CC----TCG	
A1y:	ACT**G**CCCCTCG		**A2y:**	ACT**G**CC**CCCC**TCG	
	POS	**REF**	**ALT**		
VCF	4	G	A,G		
REC1:	6	C	CCC		
VCF	4	G	A,G		
REC2:	6	C	CCCCC,C		

Figure 5.1: Example of issue with our method and random phasing. Pos: Right most digit of position in sequence. Ref: Reference sequence. VCF IN: input VCF. Phase1 and Phase2 are two different random phasing for VCF IN. A1x and A1y: two mutated alleles considering first Phase1. A2x and A2y two mutated alleles considering Phase2. VCF REC1 and VCF REC2: Recalled VCF considering Phase1 and Phase2 respectively

Finally, the reason we decided to use *vt normalize* for the input VCF is *DWGsim* that it sometimes reports indels with two preceding bases, i.e., ACT replaced with AC, while in VCF format the symbol A is unnecessary. We have seen that *GATK FARM* skips such variants without producing warnings or errors. Another issue happens when we use the *NEAT* variant simulator. With the help of *vt normalize* we noticed that the position reported for some of the simulated indels was incorrect and therefore we avoided using the *NEAT* variant simulator in our experiment. These errors are extremely rare and have almost no impact on the final comparison result. However, we decided to avoid them by using *vt normalize* on the input to maintain perfectness of the work.

6 Conclusions

Besides available variant normalization methods that process variants individually or look at small sets of variants within a narrow region of the reference sequence, based on the result represented in this paper, it is required to make use of high level VCF normalization method which recalls variants using a unique procedure. Although the best-normalized VCF file has not been defined yet, our proposed VCF normalization technique shows considerable improvement in resolving ambiguities in variant representation.

References

[1] Jason O'Rawe et al. "Low concordance of multiple variant-calling pipelines: practical implications for exome and genome sequencing." In: *Genome medicine* 5.3 (Jan. 2013), p. 28. ISSN: 1756-994X. DOI: 10.1186/gm432.

[2] Mehdi Pirooznia et al. "Validation and assessment of variant calling pipelines for next-generation sequencing". In: *Human Genomics* 8.1 (Jan. 2014), p. 14. ISSN: 1479-7364. DOI: 10.1186/1479-7364-8-14.

[3] Xiangtao Liu et al. "Variant callers for next-generation sequencing data: a comparison study." In: *PloS one* 8.9 (Jan. 2013), e75619. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0075619.

[4] Heng Li et al. "The Sequence Alignment/Map format and SAMtools." In: *Bioinformatics (Oxford, England)* 25.16 (Aug. 2009), pp. 2078–9. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btp352.

[5] *Whole Genome Simulator for Next-Generation Sequencing.* URL: https://github.com/nh13/DWGSIM (visited on 01/02/2016).

[6] "The variant call format and VCFtools." In: *Bioinformatics (Oxford, England)* 27.15 (Aug. 2011), pp. 2156–8. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btr330.

[7] Adrian Tan, Gonçalo R Abecasis, and Hyun Min Kang. "Unified representation of genetic variants." en. In: *Bioinformatics (Oxford, England)* 31.13 (July 2015), pp. 2202–4. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btv112.

[8] Aaron McKenna et al. "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data." In: *Genome research* 20.9 (Sept. 2010), pp. 1297–303. ISSN: 1549-5469. DOI: 10.1101/gr.107524.110.

[9] Mark A DePristo et al. "A framework for variation discovery and genotyping using next-generation DNA sequencing data." In: *Nature genetics* 43.5 (May 2011), pp. 491–8. ISSN: 1546-1718. DOI: 10.1038/ng.806.

[10] *Next-generation sequencing Error Analysis Toolkit.* URL: http://web.engr.illinois.edu/~zstephe2/read_simulator/ (visited on 01/02/2016).

[11] A L Delcher et al. "Alignment of whole genomes." In: *Nucleic acids research* 27.11 (June 1999), pp. 2369–76. ISSN: 0305-1048.

[12] A. L. Delcher. "Fast algorithms for large-scale genome alignment and comparison". In: *Nucleic Acids Research* 30.11 (June 2002), pp. 2478–2483. ISSN: 13624962. DOI: 10.1093/nar/30.11.2478.

[13] Stefan Kurtz et al. "Versatile and open software for comparing large genomes." In: *Genome biology* 5.2 (Jan. 2004), R12. ISSN: 1465-6914. DOI: 10.1186/gb-2004-5-2-r12.

[14] *MUMmer SNP format to VCF format converter*. URL: https://github.com/douglasgscofield/bioinfo/blob/master/scripts/mummer2Vcf.pl (visited on 01/02/2016).

[15] Heng Li. "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM". In: (Mar. 2013), p. 3. arXiv: 1303.3997.

[16] *Picard Tools*. URL: http://broadinstitute.github.io/picard (visited on 01/02/2016).

[17] Zhi Wei et al. "SNVer: a statistical tool for variant calling in analysis of pooled or individual next-generation sequencing data." In: *Nucleic acids research* 39.19 (Oct. 2011), e132. ISSN: 1362-4962. DOI: 10.1093/nar/gkr599.