Alignment-Free Genotyping of Known Variations with MALVA

Giulia Bernardini, Luca Denti, and Marco Previtali

Abstract

The discovery and characterization of sequence variations in human populations are crucial in genetic studies. Standard methods for addressing this problem are computationally expensive and highly time consuming, thus impractical for clinical applications, where time is often an issue. When the task is to genotype variations that have been previously annotated, alignment-free methods come to the aid. Here, we describe **MALVA**, an alignment-free approach for genotyping a set of known variations. **MALVA** is the first mapping-free tool which is able to genotype multi-allelic SNPs and indels, even in high-density genomic regions, and to effectively handle a huge number of variations.

Key words Variant genotyping, Mapping-free, *k*-mers, Next-generation sequencing, Bioinformatics

1 Introduction

Current approaches to call variations from NGS samples can be summarized in three categories. Alignment-based approaches, such as **GATK** [1] and **BCFtools** [2], that first align the reads against a reference genome and infer genotypes from the alignments; assembly-based approaches, such as **DiscoSNP++** [3], that assemble the reads in an assembly graph and then detect variations by analyzing such graph; and variation graph-based approaches such as **GraphTyper** [4] and **vg** [5]. However, all these approaches are highly time consuming, and they can be hardly applied in medical contexts, where it is necessary to be accurate and extremely fast.

When the organism under investigation is well characterized, it is possible to reduce the task of variant calling to the task of variant genotyping, that is, to call the genotype only in those positions where variants have been previously annotated. Variant genotyping is particularly relevant to medical applications, which often require to know the genotype at certain loci that are known to be of medical importance.

The task of genotyping known variants has been recently addressed by alignment-free methods that genotype variations directly from raw sequencing reads, i.e., without mapping them to the reference genome [6–8]. Such approaches are extremely fast and accurate, but they have large memory requirements, which can easily exceed hundreds of GB of RAM. An even more significant limitation of such tools is that they focus on biallelic SNPs, providing very limited support for multi-allelic SNPs and short insertions and deletions of nucleotides (indels).

MALVA is an alignment-free tool that addresses all the limitations of previous alignment-free approaches: it is a lightweight and fast method, which is effectively able to handle close and multi-allelic SNPs as well as indels. At a high-level, the peculiarities of **MALVA** are the following.

1. It represents each allele of a variation using a set of k-mers (length-k substrings), allowing to easily model multi-allelic variations, i.e., those variations for which multiple alleles are known (different individuals may express a different allele for the same variation).
2. It allows for modeling and genotyping indels, again in virtue of using a set of k-mers instead of a single k-mer for representing variations.
3. By using lightweight and efficient data structures, it is extremely fast and requires little memory.
4. It uses k-mer counting to extract information from the input raw NGS sample, thus making it extremely fast if compared to alignment-based approaches.
5. It uses a multinomial distribution (instead of the binomial distribution used by previous methods) to call the genotypes of known variations, allowing for genotyping multi-allelic variations.

2 Materials

MALVA is distributed under the GPL-3.0 open-source license, and the source code is available at https://github.com/algolab/malva. **MALVA** is also available on the Bioconda repository, and therefore, it can be easily installed via the `conda` package management system.

2.1 Software

MALVA was developed in C++, and it has been developed and tested on 64bit Linux system. Indeed, **MALVA** cannot be run on MacOS or Windows system due to dependencies incompatibility.

MALVA depends on the following tools and packages:

1. C++11-compliant compiler. We recommend using GCC 7.4 or newer.
2. **sdsl-lite** library (v2.1.1) [9], a C++11 library implementing succinct data structures. **MALVA** uses this library to implement Bloom filters, which constitute its core data structure.
3. **htslib** (v1.10.2 or newer), a C library for accessing common file formats used for high-throughput sequencing data. This library is required to read and parse the input VCF file containing the set of known variations.
4. **KMC** (v3.1.1 or newer), an efficient *k*-mer counter. This is used to perform the *k*-mer counting step of the **MALVA** pipeline.

2.2 Download and Install MALVA

The source code of **MALVA** is freely available at https://github.com/algolab/malva. The latest version of **MALVA** can be retrieved using the following command:

```
git clone --recursive https://github.com/AlgoLab/malva.git
```

For convenience, all the required dependencies are included in the repository. If **sdsl-lite**, **htslib**, and **KMC3** are not already installed, they can be installed by running the following commands:

```
cd sdsl-lite
./install.sh ..
cd ../KMC
make
cd ../htslib
make
cd ..
```

Finally, **MALVA** can be compiled from the root of the local copy of the repository simply by running:

```
make
```

To simplify its installation, **MALVA** is also available on Bioconda. Therefore, once the `conda` package management system is installed and the Bioconda channel is enabled, **MALVA** and all its dependencies can be installed with a single command:

```
conda install malva
```

3 Methods

The alignment-free approach implemented by **MALVA** can be summarized in the following three steps (also illustrated in Fig. 1). Further details can be found in the supplemental material of [10].

1. Each input variation is characterized by two sets of strings of fixed lengths $k < r$, respectively: the shorter k-mers are used to represent the alleles of the variation, whereas the longer r-mers give information about the genomic region surrounding the variation's locus. Intuitively, the longer r-mers are used to tell apart the variants that are actually expressed at some locus from other genome repetitions. Both sets are stored in a Bloom filter.
2. All r-mers are extracted from the input raw read sample along with their frequencies (i.e., the number of times they occur in the sample). These frequencies are then used to assign a weight to each of the k-mers computed in **step 1**.
3. The input variants are genotyped using a probabilistic framework based on the Bayes' theorem that combines allele (a priori) frequencies and k-mers weights.

3.1 Inputs, Outputs, and Parameters

To see required and optional parameters for **MALVA**, run:

```
MALVA -h
```

Indeed, to ease the execution of **MALVA**, we provide a bash script (named `MALVA`) that runs the entire **MALVA** pipeline. The required input files to run **MALVA** successfully are:

1. The reference genome in FASTA format.
2. The set of know variations of the population under investigation in VCF format (*see* **Notes 1** and **2**).
3. A NGS read sample in FASTA/Q format.

Therefore, the command

```
MALVA /path/to/genome /path/to/vcf /path/to/sample
```

runs the **MALVA** pipeline and prints to standard output a new VCF file containing the input variations genotyped with respect to the input sample. For the sake of convenience, we suggest to redirect **MALVA** output to file using output redirection:

```
MALVA /path/to/genome /path/to/vcf /path/to/sample > /path/to/
output/vcf
```

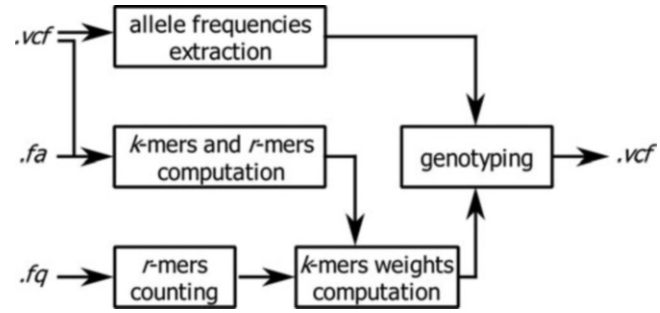

Fig. 1 The approach of MALVA. Input files are the reference genome (FASTA format) along with known variations (VCF format) and a raw reads sample (FASTQ format). First, known alleles are characterized using k-mers and r-mers. Then, r-mer counting from input sample is used to assign a weight to each r-mer (and k-mer). Finally, the allele frequencies and the k-mer weights are used to genotype the input variation, producing a new VCF file

Optional parameters can be tweaked to adapt the execution of **MALVA** to user needs:

- `-k <int>` and `-r <int>`: **MALVA** characterizes each input variation by two sets of strings of different fixed lengths `k` < `r` (*see* **Note 3**). The default values are 35 and 43, respectively. In principle, larger values of `k` and `r` improve the precision of the results while making the method computationally heavier, although using longer `k`-mers will increase the probability of covering erroneous bases in them. The default values are optimized for NGS data.
- `-e <float>` is the expected error rate of the input sample. The default value is 0.001.
- `-s <filepath>` is a file containing the samples of the input VCF to consider. Use this if you want to genotype the set of variations with respect to a subset of the population used to build the VCF. By default, **MALVA** considers all the samples contained in the input VCF (*see* **Note 4**).
- `-f <string>` is the name of the INFO field of the VCF containing the allele frequency. The default value is `AF` but users should set this value in accordance to the name of the field containing the allele frequency in their VCF (*see* **Note 5**).
- `-c <int>` is the maximum coverage allowed per allele. Each variation with alleles that are covered too much (more than `c`) is genotyped as 0/0, as extremely high coverages are a hint of genome repetitions. In these cases, **MALVA** is conservative and prefers not to genotype the variation. The default value is 200. Larger values of `c` lead, in general, to a larger recall, while lowering the method's precision.

- `-b <int>` is the size in GigaByte of the Bloom filter used to store the `k`-mers (*see* **Note 6**). The default value is 4. Larger values of `b` lower the Bloom filter's false-positive rate but make the method more memory-intensive.
- `-m <int>` is a parameter of KMC. It indicates that KMC will attempt to use a maximum amount of `m` GigaBytes during `k`-mer counting, though the bound could be exceeded in the case of very large datasets. The default value is 4.
- `-u` tells **MALVA** to use uniform `a priori` probabilities. By default, **MALVA** derives such probabilities from the allele frequencies provided by the input VCF (*see* **Note 5**).
- `-1` tells **MALVA** to run in haploid mode. The default **MALVA** mode is diploid.

3.2 Interpreting the Output File

The output of **MALVA** is a VCF file containing the input variations that have been genotyped and their genotype, computed with respect to the input sample of raw reads. For each of the considered variations (*see* **Note 7**), **MALVA** reports the most likely (unphased) genotype along with its likelihood. For example, consider the following variation (a line from the output of **MALVA**):

```
20 66370 rs6054257 G A,T 100 PASS . GT:GQ 0/2:100
```

There are five relevant fields in this record. It describes a multi-allelic SNP occurring on chromosome 20 (first field) at position 66370 (second field) with reference allele G (fourth field) and two possible alternative alleles, A and T (fifth field). **MALVA** genotyped such SNP as 0/2 with likelihood 100% (last field). Since this is a multi-allelic variation, **MALVA** computed six different genotypes (i.e., 0/0, 0/1, 0/2, 1/1, 1/2, and 2/2) and then produced as output the one with the highest likelihood.

3.3 Running MALVA on Humans (Diploid Organism)

Let us assume we have a human sample we want to genotype. Since **MALVA** relies on an input set of variations to be efficient and accurate, it is necessary to retrieve such a set before running **MALVA**. If the sample under investigation is from a human individual, the VCF provided by the 1000 Genomes Project is a good candidate, especially when the individual under analysis is from one of the populations targeted by the project (*see* **Note 4**). Unfortunately, the 1000 Genomes Project provides a single VCF per chromosome. Therefore, once downloaded, it is necessary to concatenate them into a single VCF, e.g., using `bcftools concat` [11]. Finally, **MALVA** needs an input reference genome: assuming we have downloaded the VCFs produced by mapping reads against GRCh37, that will do (*see* **Note 2**).

Once we have all the required input files, we can genotype the individual under investigation with the following command:

```
MALVA /path/to/GRCh37.fa /path/to/1kgp.vcf /path/to/human-sample.fq > genotypes.vcf
```

Although MALVA is already fast (*see* **Note 8**), additionally, to speed up its computation while (possibly) lowering the overall genotype accuracy, we can limit **MALVA** to only consider the annotated genotypes of the individuals that belong to the same population as the sample under investigation (*see* **Note 4**). To do so, we need to select such individuals and produce a text file containing their IDs (as appearing in the VCF file), one name per line. In the case of the 1000 Genomes Project, it is possible to easily produce this file using the sample panel provided along with the VCFs. For instance, once downloaded the panel file (`integrated_call_samples_v3.20130502.ALL.panel`) associated with the phase 3 callset provided by the 1000 Genomes Project, it is possible to extract the list of European samples using command-line utilities usually available on any unix-based system:

```
grep "EUR" integrated_call_samples_v3.20130502.ALL.panel | cut -f 1 > eur.panel
```

We can then run **MALVA**:

```
MALVA -s eur.panel /path/to/GRCh37.fa /path/to/1kgp.vcf /path/to/human-sample.fq > genotypes.vcf
```

3.4 Running MALVA on Viruses (Haploid Organism)

As stated above, one of the most important prerequisites of **MALVA** is the creation or retrieval of the input set of variations to be genotyped. However, when the sample under investigation is not from a well-characterized species, a VCF of known variations may not be always readily available. This is the case, for example, of viruses. Even in this case, though, it is still possible to take advantage of the efficiency of **MALVA**. Indeed, if multiple complete assemblies of the species of interest are available, it is possible to create a VCF file of known variations by multi-aligning the assemblies and then extracting the variations from the multiple alignments. This is a common and simple pipeline, but the user must implement it by himself. Alternatively, it is possible to exploit more sophisticated pipelines, such as those proposed and implemented in **MALVIRUS** [12], that relieve the user from this burden and provide a simple to use interface. Differently from humans, that are diploids, viruses are haploids, meaning they have a single haplotype per chromosome. Therefore, in this new scenario, the genotyping module of **MALVA** must be tweaked accordingly: the

genotypes to compute are not 0/0, 0/1, and so on, but they are simply 0, 1, 2... Switching the genotyping module to the haploid one is as easy as adding the `-1` argument to **MALVA** command:

```
MALVA -1 /path/to/reference.fa /path/to/variations.vcf /path/
to/virus-sample.fq > genotypes.vcf
```

4 Notes

1. The input VCF must contain the set of variations observed in a population along with the observed genotypes (*see* for example the VCF files provided by the 1000 Genomes Project[1]). Indeed, **MALVA** leverages on the known genotypes to build the *k*-mers associated with each allele. For this reason, **MALVA** cannot work with VCF containing variations only (such those provided by dbSNP).
2. The reference genome and the input VCF must use the same naming conventions (i.e., if the names of the chromosome start with `chr` in the FASTA file, they must start in the same way in the VCF as well). Moreover, the reference genome must be the same as the one used to produce the VCF (i.e., if the input calls were called against GRCh38, then GRCh38 must be used as reference genome).
3. As previously mentioned, **MALVA** identifies each variant allele with a set of *k*-mers, for a fixed value of *k* chosen by the user. Roughly speaking, each such *k*-mer is the *k*-long substring centered in the variant that would occur in a genome in which that allele is expressed (possibly in combination with other close variants). **MALVA** then calls the genotypes by carefully analyzing the frequency in the input sample of each *k*-mer describing variants alleles. Choosing a small *k* makes the method less computationally heavy, but the smaller the value of *k*, the more probable that the same *k*-mer occurs at different locations of the genome, potentially leading **MALVA** to call the wrong genotype due to these repetitions. To avoid most of such wrong calls, when a *k*-mer associated to a variant allele is repeated in other places of the reference genome, **MALVA** makes use of a larger context around the allele to distinguish it from the other occurrences. Contexts are identified by longer *r*-mers, whose length $r > k$ can also be tuned by the user. Since, in general, there are fewer contexts to be checked then variants to genotype, in this way **MALVA** achieves a convenient trade-off between memory usage and precision. For technical

[1] ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/

reasons, it is advisable that both r and k are chosen to be odd numbers.

4. Since **MALVA** genotypes a new sample using a VCF built from a population of individuals of the same species, it is possible to speed the computation up by subsampling the population or by directly considering a subpopulation. For instance, if the sample we are interested in is from a European individual, we may consider genotyping it with respect to the other European individuals contained in the input VCF only. For this reason, **MALVA** provides the `-s` parameter. However, focusing on a single (sub)population may lower the overall accuracy of **MALVA**, especially in the event of contaminations among different (sub)populations.
5. To better genotype input variations, **MALVA** combines the information contained in the VCF (i.e., the frequency of each allele of a variation), and the information extracted by the read sample (i.e., the coverage of each allele). Typically, the allele frequencies are contained in the `INFO` column of the VCF file, encoded using the `AF` key. If a VCF encodes this information with a different key, it is possible to set the new key using the `-f` parameter. Otherwise, if a VCF does not contain this information, it is possible to use uniform frequencies using the `-u` parameter.
6. **MALVA** stores sets of k-mers in two Bloom filters, which are very efficient probabilistic data structures that allow membership queries. The size of a Bloom filter is fixed, and it is decided before its creation: a bigger Bloom filter should reduce the probability of false positives in membership queries. **MALVA** uses by default two Bloom filters of 4 GB each, thus requiring at least 8 GB RAM (plus overhead). However, users must set the corresponding parameter to suit their needs: if 4 GB is too high a requirement, we have experimentally assessed that, when the task is to genotype known human variations, Bloom filters of 2 GB each are large enough to produce reasonably good results.
7. **MALVA** genotypes all the variations contained in the input VCF except those variations with symbolic alleles (e.g., non-nucleotides alleles such as `<INS>` and `<DEL>`).
8. **MALVA**, thanks to its mapping-free nature, is very fast: it genotyped the full callset provided by the 1000 Genomes Project (2504 individuals and 84,739,838 variants) from a 30× human sample (696168435150-bp-long reads) in less than 10 h and using 39 GB RAM. For comparison, **GATK** and **BCFtools**, two widely used alignment-based approaches, completed their analysis in more than 2 and 3 days requiring 9 GB and 33 GB RAM, respectively.

References

1. McKenna A, Hanna M, Banks E et al (2010) The genome analysis toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res 20(9): 1297–1303
2. Li H (2011) A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. Bioinformatics 27(21):2987–2993
3. Peterlongo P, Riou C, Drezen E, Lemaitre C (2017) DiscoSnp++: de novo detection of small variants from raw unassembled read set (s). BioRxiv
4. Eggertsson HP, Jonsson H, Kristmundsdottir S et al (2017) Graphtyper enables population-scale genotyping using pangenome graphs. Nat Genet 49:1654–1660
5. Garrison E, Sirén J, Novak AM et al (2018) Variation graph toolkit improves read mapping by representing genetic variation in the reference. Nat Biotechnol 36(9):875–879
6. Pajuste FD, Kaplinski L, Möls M et al (2017) FastGT: an alignment-free method for calling common SNVs directly from raw sequencing reads. Sci Rep 7(1):1–10
7. Shajii A, Yorukoglu D, William YY, Berger B (2016) Fast genotyping of known SNPs through approximate k-mer matching. Bioinformatics 32(17):i538–i544
8. Sun C, Medvedev P (2019) Toward fast and accurate SNP genotyping from whole genome sequencing data for bedside diagnostics. Bioinformatics 35(3):415–420
9. Gog S, Beller T, Moffat A, Petri M (2014) From theory to practice: plug and play with succinct data structures. In: International symposium on experimental algorithms. Springer, Cham, pp 326–337
10. Denti L, Previtali M, Bernardini G et al (2019) MALVA: genotyping by mapping-free ALlele detection of known VAriants. IScience 18:20–27
11. Danecek P, Bonfield JK, Liddle J et al (2021) Twelve years of SAMtools and BCFtools. Gigascience 10(2):giab008
12. Ciccolella S, Denti L, Bonizzoni P et al (2020) MALVIRUS: an integrated web application for viral variant calling. BioRxiv