Bioinformatics, 37(4), 2021, 568–569
doi: 10.1093/bioinformatics/btaa716
Advance Access Publication Date: 11 August 2020
Applications Note

Genome analysis

Mutation-Simulator: fine-grained simulation of random mutations in any genome

M. A. Kühl, B. Stich and D. C. Ries*

Quantitative Genetics and Genomics of Plants, Heinrich Heine University, Düsseldorf 40225, Germany

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on March 18, 2020; revised on June 12, 2020; editorial decision on July 6, 2020; accepted on August 5, 2020

Abstract

Summary: Mutation-Simulator allows the introduction of various types of sequence alterations in reference sequences, with reasonable compute-time even for large eukaryotic genomes. Its intuitive system for fine-grained control over mutation rates along the sequence enables the mimicking of natural mutation patterns. Using standard file formats for input and output data, it can easily be integrated into any development and benchmarking workflow for high-throughput sequencing applications.

Availability and implementation: Mutation-Simulator is written in Python 3 and the source code, documentation, help and use cases are available on the Github page at https://github.com/mkpython3/Mutation-Simulator. It is free for use under the GPL 3 license.

Contact: davidries@protonmail.com

1 Introduction

The benchmarking of bioinformatics programs for read alignment, variant and RNA-Seq analysis as well as genome assembly is frequently performed on the basis of simulated genomic sequence datasets (Mangul et al., 2019; Price and Gibas, 2017). The advantage of such an approach is that it allows for the systematic assessment of performance even if no gold-standard data is available (Mangul et al., 2019). Many tools exist for the generation of simulated reads from a reference sequence (for review, see Escalona *et al.*, 2016).

However, for the benchmarking of bioinformatics programs for the identification of structural variation (SV) and for genome assembly, read simulators can only be applied to reference sequences after mutations have been simulated for this sequence. The capacity to introduce SV of any size, as well as genomic rearrangements is a feature needed to evaluate tools using long-read, mate-pair or chromosome conformation-capture sequencing.

To the best of our knowledge, the programs described for such a simulation of mutations are SIM-CT (Audoux *et al.*, 2017), SVsim (https://github.com/GregoryFaust/SVsim), shuffleseq (http://emboss.sourceforge.net/apps/release/6.2/emboss/apps/shuffleseq.html), simuG (Yue and Liti, 2019) and Simulome (Price and Gibas, 2017). Compared to Mutation-Simulator, these tools are either limited in the type of genomes they can operate on (Simulome), limited in their field of use (e.g. SimCT), much slower (simuG) or only allow for the simulation of less mutation types and lack fine tuning options (e.g. SVsim and shuffleseq). With a combination of all features implemented in Mutation-Simulator, users can create realistic mutational patterns for all types of genomes with a single tool.

2 Features and methods

Mutation-Simulator is a versatile and intuitive open source Python program featuring a wide range of mutation types as well as transition/transversion rates to simulate mutations in reference genomes. It provides a coherent description of the implemented mutations in standardized formats such as variant call format (VCF) and BEDPE (https://bedtools.readthedocs.io/en/latest/content/general-usage.html#bedpe-format), allowing for easy integration with a wide range of bioinformatics tools. Mutation-Simulator can utilize the new file format Random Mutation Tables (RMT) to achieve a fine-grained control over simulated mutation patterns. It operates with FASTA files as input and inserts SNP and SV according to user-defined probabilities. Mutation-Simulator features the following SV types: insertions, deletions, inversions, tandem duplications, translocations and interchromosomal translocations.

It provides three different modes to generate a simulated genome. ARGS mode can be used to quickly generate a mutated genome with uniform mutation rates across the genome. IT mode simulates interchromosomal exchange of DNA sequences. RMT mode is a combination of ARGS and IT mode, with the additional possibility to define specific mutation rates for each region of the genome.

The RMT mode alongside the new RMT file format is the core feature of Mutation-Simulator, while the IT and ARGS mode exist

as quality of life features if creating a RMT file for a given use case is unnecessary.

In any mode, the user specifies an existing genome and mutation rates of the desired types which will then be used to generate mutations across the reference genome. Depending on the used mode, the user has the option to block bases after each mutation or whole genomic regions, define mutation rates that vary throughout the genome and set the maximum length for every mutation type. Simulations with Mutation-Simulator are not limited to the size or type of the reference genomes. This allows users to simulate versions of every desired genome, regardless of whether they are prokaryotic or eukaryotic. For details on how to run each mode, please consult the Github page or the Mutation-Simulator's help function.

2.1 ARGS mode

ARGS allows to simulate uniformly distributed SNP or any kind of SV with a defined rate and maximum length across the reference sequences.

2.2 IT mode

The IT mode is used to simulate reciprocal interchromosomal translocation events. For this purpose, two chromosomes are randomly chosen as pairs. Thereupon breakend positions are generated randomly and independently on both chromosomes at a specified rate. The sequences downstream of these positions are then exchanged between the pair of chromosomes.

2.3 RMT mode

RMT mode is a combination of the aforementioned modes with the additional possibility to define specific mutation rates for each region of the genome. The user can define desired mutation types, lengths and rates for each region separately instead for the whole genome. RMT mode also allows for specification of IT rates for each chromosome, set standard values for unspecified regions and block positions entirely from mutations. This is achieved by providing the required information in a RMT file.

The RMT file has three sections: 'Meta-Information' (optional), 'Standard' and 'Range-Definitions' (optional). 'Meta-Information' allows for noting optional information and check marks for the sequence or project. The 'Standard' section contains specifications about mutation rates and applies when no range definitions are set, or whenever ranges or chromosomes are missing. The 'Range-Definitions' are the key feature of the RMT file format and Mutation-Simulator. It allows for defining areas of the genome where the mutation rates differ from the one defined in the 'Standard' section. An easy way to automatically create RMT files to mimic *in vivo* mutation distributions from existing data of a species (e.g. from VCF or GTF files) is to count the occurrences of mutations in bins along a sequence. A detailed description of the workflow on how to create RMT files alongside species-specific example RMT files for common model organisms is provided in the documentation.

2.4 Output formats

The mutated genome is presented as a standard multi-FASTA file. In addition, the changes made to the input genome are stored in VCF version 4.3. Since the VCF format is not suitable to describe multiple large genomic rearrangements, these are stored in BEDPE format. These two are the most widely used formats to describe mutations, which facilitates the integration of Mutation-Simulator in existing workflows.

3 Performance

To evaluate speed and memory consumption, we tested Mutation-Simulator in ARGS mode on single sequences of increasing sizes with the following parameters:

–sn 0.01 -in 0.01 -de 0.01 -du 0.01 -iv 0.01 -tl 0.01 -inl 3
–del 3 -dul 3 -ivl 3 -tll 3

which correspond to high mutation rates compared to the spontaneous mutation rate of 7×10^{-9} base substitutions per site per generation that was observed for *Arabidopsis thaliana* (Ossowski *et al.*, 2010).

The tests were run on the Debian 9 operating system with an Intel Xeon CPU E5-4660 v4 @ 2.20 GHz. For sequence sizes of 1, 10, 100 Mbp and 1 Gbp, Mutation-Simulator took less than 7, 75, 700 and 6000 s to finish, with a peak memory consumption of 89, 643, 6025 and 58 911 MB, respectively. The upper limit of memory consumption is affected by the number of introduced mutations as well as the length of the largest mutated chromosome.

A direct comparison of Mutation-Simulator to Simulome and SVsim using an *E.coli* K-12 genome showed that Mutation-Simulator performed 24 and 207 times faster, with only 34.66 MB and 83.64 MB more RAM usage, respectively. Comparing Mutation-Simulator's performance to simuG on inserting 10×10^{3} SNPs and INDELs in a 10 Mb test sequence resulted in Mutation-Simulator being 6785 times faster. For more details, please refer to the Github page.

Funding

This research was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy—EXC 2048/1—Project ID: 39068611.

Conflict of Interest: none declared.

References

Audoux,J. *et al.* (2017) SimBA: a methodology and tools for evaluating the performance of RNA-Seq bioinformatic pipelines. *BMC Bioinformatics*, **18**, 428.

Escalona,M. *et al.* (2016) A comparison of tools for the simulation of genomic next-generation sequencing data. *Nat. Rev. Genet.*, **17**, 459–469.

Mangul,S. *et al.* (2019) Systematic benchmarking of omics computational tools. *Nat. Commun.*, **10**, 1–11.

Ossowski,S. *et al.* (2010) The rate and molecular spectrum of spontaneous mutations in Arabidopsis thaliana. *Science*, **327**, 92–94.

Price,A. and Gibas,C. (2017) Simulome: a genome sequence and variant simulator. *Bioinformatics*, **33**, 1876–1878.

Yue,J. and Liti,G. (2019) simuG: a general-purpose genome simulator. *Bioinformatics*, **35**, 4442–4444.