vcfsim: flexible simulation of all-sites VCFs with missing data

Paimon Goulart[1], Kieran Samuk[1]

[1]Department of Evolution, Ecology, and Organismal Biology University of California Riverside

Abstract

VCFs are the most widely used data format for encoding genetic variation. By design, standard VCFs do not include data from sites where all individuals are homozygous for the reference allele ("invariant sites") and thus do not differentiate these from sites where data are completely missing. However, missing data are a key feature of biological datasets across all domains of genomics, and many recent studies have shown that missing data can introduce a variety of statistical biases in the estimation of key population genetic parameters. A solution to this limitation is to include invariant sites in a standard VCF, creating an "all-sites VCF", exposing missing and invariant sites explicitly. One hurdle to the wider adoption of all-sites VCFs is a reliable parameterized simulation framework for generating biologically realistic all-sites VCFs. Here, we introduce an open-source command line tool, *vcfsim*, that interfaces with the popular coalescent simulation platform *msprime* and provides convenience functions for simulating all-sites VCFs with variable levels of ploidy and missing data. We show that the post-processed VCFs generated using *vcfsim* align precisely with population genetic expectations (i.e. are statistically identical to raw *msprime* output) and can accurately introduce missing data and varying ploidy levels, including the simulation of intraindividual ploidy variation (e.g. heterogametic sex chromosomes). We suggest *vcfsim* will be a useful tool for the benchmarking of new software tools, training of machine learning models, and the exploration of the effects of missing data in genomics data sets.

Introduction

The Variant Call Format (VCF) is the most widely used file format for encoding genetic variation data, particularly in the context of population genetics and genomics studies (Danecek et al., 2011). VCF files are designed to store information about genetic variants, including single nucleotide polymorphisms (SNPs), insertions, deletions, and structural variants. Despite their widespread usage, VCFs have several key limitations. One significant drawback is that, by design, sites where all individuals are homozygous for the reference allele, i.e. invariant sites, are typically omitted (Danecek et al., 2011). This feature makes it impossible to distinguish between sites that are missing due to lack of sequencing data and sites that are simply invariant. This missing information complicates downstream analyses, particularly in cases where distinguishing between these types of sites is crucial, such as the estimation of nucleotide diversity within and between populations (Korunes & Samuk, 2021; Samuk, 2023).

One common workaround is the use of "all-sites" VCFs, which include both variant and invariant sites. This approach is often used in tools like *pixy* (Korunes & Samuk, 2021), which makes use of

all-sites VCFs to calculate population genetics statistics while accounting for missing data and per-site differences in sample size. However, benchmarking tools that make use of all-sites VCFs has been challenging due to the lack of native support from common simulation packages, which typically output standard (i.e. variants only) VCFs. Simulating all-sites VCFs typically involves complex and ad-hoc workflows, especially when missing data and other biological details are involved.

Indeed, missing data are widely known to introduce serious biases in the estimation of a variety of population genetic parameters (Korunes & Samuk, 2021; Lowry, 2017; Samuk, 2023; Wong, Zeng, & Lin, 2019), yet there are no widely adopted tools for simulating ground-truth data with predefined levels of missing genotypes or sites. This is an issue, as researchers often need simulated data with specific patterns of missing information or population genetic parameters to validate tools or analyses, but current methods require manual intervention and lack flexibility. As such, there is a need for standardized tools to simulate population genetic data and output all-sites VCFs.

Here, we introduce *vcfsim*, a command-line tool that provides a user-friendly interface to the coalescent simulation package *msprime* (Baumdicker et al., 2022). *vcfsim* allows for the generation of all-sites VCFs from coalescent simulations with customizable levels of missing data. The tool also supports simulation of VCFs across various ploidy levels, including mixed ploidies (e.g., diploid organisms with haploid sex chromosomes), making it a versatile solution for researchers needing flexible and accurate simulated genetic data. By addressing the limitations of current tools, *vcfsim* simplifies the process of creating biologically realistic VCFs for use in testing and validating genetic analysis pipelines.

Materials and Methods

We wrote *vcfsim* in Python version 3.6 as well as the libraries: ipython (Perez and Granger 2007), numpy (Harris et al. 2020), and pandas (The pandas development team 2024). The coalescent simulation package msprime (Baumdicker et al., 2022) provides the core population genetic-based backend for the simulation. The source code for *vcfsim* can be found at https://github.com/Pie115/*Vcfsim*ulator-SamukLab. It can also be easily installed via the package management platform conda (conda contributors 2025), on the bioconda (Grüning et al. 2018) channel https://bioconda.github.io/recipes/*vcfsim*/README.html.

vcfsim simulates realistic genetic variation data that can be highly customized to match a wide variety of use cases. The tool works by simulating genetic data from a single population using a coalescent simulation process (via msprime), applying mutations to these sequences, outputting the results in VCF format, and post-processing this VCF to simulate missing data of various kinds. Users can customize the simulation via command-line arguments to specify the desired characteristics of the simulated population and VCF output. Key features of *vcfsim* include:

- Random seed control: Users can specify a random seed to control stochastic elements of the coalescent simulation and missing data processes. Identical seeds yield identical results for repeated runs, ensuring reproducibility.
- Missing data simulation: The software allows for the simulation of missing data, both at the level of genotypes and whole sites.
- Flexible chromosome inputs: Users can simulate multiple chromosomes simultaneously by providing an input file with parameters for each chromosome (e.g. multiple autosomes and a sex chromosome).
- Efficient handling of large simulations: *vcfsim* supports running multiple simulation replicates from a single command, facilitating use for statistical validating and model training.

Parameters and inputs

vcfsim offers a range of user-specified customization options. These parameters include both required and optional arguments that control various aspects of the simulation process, such as the random seed for reproducibility, the level of missing data, and the characteristics of the population being simulated. The full list of required and optional parameters and given in Table 1 and Table 2 respectively.

Table 1 | Required parameters for *vcfsim*. Each row represents a single required parameter, with its name, value (in capital letters), and type of parameter (e.g. an integer, string, etc.). Square brackets indicate placeholders for actual parameter values. The description column contains a description of the function of each parameter in controlling the behavior of *vcfsim*.

Parameter	Description
--seed [SEED, integer]	The random seed used by *vcfsim* ensures reproducibility. By specifying a seed, users can generate the same VCF output in subsequent runs, which is particularly useful for debugging or comparing results.
--percent_missing_sites [PERCENT_MISSING_SITES, integer]	This argument determines the percentage of sites in the VCF that will be marked as missing. For example, if the value is set to 10, 10% of the VCF sites will be randomly selected to be missing. This feature helps simulate incomplete or imperfect genetic datasets, which are common in real-world scenarios.

--percent_missing_genotypes [PERCENT_MISSING_GENOTYPES, integer]	This controls the percentage of genotypes that are missing from the VCF for each sample. Missing genotype data is another realistic element of genetic studies, and this parameter helps simulate the random absence of genetic information across samples.
--sample_size [SAMPLE_SIZE, integer]	The number of samples to be included in the VCF output. A sample represents an individual from the simulated population, and this parameter determines how many individuals' genetic data will be simulated. Larger sample sizes will result in more data being included in the VCF.

Table 2 | Optional parameters for *vcfsim* (i.e. those not required for a run). Each row represents a single required parameter, with its name, value (in capital letters), and type of parameter (e.g. an integer, string, etc.). Square brackets indicate placeholders for actual parameter values. The description column contains a description of the function of each parameter in controlling the behavior of *vcfsim*.

Parameter	Description
--chromosome [CHROMOSOME] Default = 1	This specifies the chromosome name that the simulation will focus on. Users can define this when simulating specific chromosomes, and the result will be named accordingly in the output.
--replicates [REPLICATES] Default = 1	The number of times *vcfsim* should run the simulation. For example, setting replicates to 3 would result in three separate VCF files being generated, each with a different random seed.
--sequence_length [SEQUENCE_LENGTH] Default = 10000	The length of the genetic sequence being simulated in base pairs. This controls how many base pairs are simulated for each site, affecting the overall size of the VCF. Larger sequence lengths will generate more genetic variation and data.

--ploidy [PLOIDY] Default = 2	This parameter sets the ploidy level for the simulation, meaning the number of sets of chromosomes per individual. For example, humans have a ploidy of 2 (diploid), meaning two sets of chromosomes, one from each parent. Other organisms may have different ploidy levels.
--Ne [NE] Default = 1700000	The effective population size of the simulated population. This parameter impacts the genetic diversity of the population, with smaller populations typically having less genetic variation than larger populations.
--mu [MU] Default = 0.0000000055	This represents the mutation rate in the simulated population, defining the probability that a base pair will mutate. A higher mutation rate will result in more genetic differences among individuals, which may be useful for studying genetic diversity.
--output_file [OUTPUT_FILE] Default = myvcftest	The name of the output file where the simulated VCF will be saved. If this argument is not specified, *vcfsim* will print the output to the command line instead. When multiple replicates are generated, the file name will include the seed value to distinguish between runs. If an output file isn't specified, then *vcfsim* will automatically output the vcf to the command line.
--param_file [PARAM_FILE] Default = None	This optional argument allows users to input a file that specifies multiple parameters for simulating different chromosomes. Each row in the input file represents a separate simulation, and *vcfsim* will concatenate the results from each run into a single output file. This feature is particularly useful for large-scale simulations involving multiple chromosomes or different parameter sets.

Output options

vcfsim provides flexibility in how users manage their output. Users can specify an output file for the generated VCF data using the "--output_file" argument. The resulting file will always be in the .vcf format and will append the random seed used for the simulation at the end of the filename, ensuring repeatability of results. If no output file is specified, the VCF data will be printed directly to the command line (i.e. standard output). This feature allows users to integrate *vcfsim* easily into command line pipelines, enabling further processing or analysis without the need to generate a file.

Validation of vcfsim

In order to test the accuracy of *vcfsim*, we performed a variety of validation procedures.The accuracy of *msprime* has been well demonstrated (Baumdicker et al., 2022)*, however, vcfsim* performs substantial post-processing of *msprime* VCFs via the addition of invariant sites and missing data, multiple chromosomes, and variable ploidy. As such, we sought to validate the following:

First, to evaluate whether the simulated VCFs generated by *vcfsim* align with theoretical expectations of nucleotide diversity (π), we conducted a large-scale simulation. We generated 10,000 diploid VCF files, each with a mutation rate of 1e-9, an effective population size (N_e) of 1,700,000, a sequence length of 10,000 base pairs, and a sample size of 10 (i.e. 10 diploid genotypes). Each simulation was performed with a unique random seed, starting from 1 000 and incrementing by 1 up to 11 000. To measure π for each simulated VCF, we employed both VCFtools (Danecek et al., 2011) and pixy (Korunes & Samuk, 2021) to ensure consistency between the tools and to verify that *vcfsim* did not introduce any errors. This dual approach allowed us to cross-check the π values calculated by both libraries.

To verify that the percent of missing genotypes and missing sites introduced by *vcfsim* align with the specified input values, we obtained an independent validation of the extent of missing data using bcftools (Danecek et al., 2021). For missing genotypes, we generated VCF files with a sample size of 100 and incrementally introduced missing genotypes, ranging from 0% to 100%, increasing by 1% with each iteration. Similarly, to assess the accuracy of missing site generation, we created another script to manually count the number of missing rows in the VCF files. In this case, we set the sequence length to 100 and progressively introduced missing sites, again in 1% increments from 0% to 100%. Ideally, we expect a perfect linear relationship between the specified and measured values, confirming that *vcfsim* is accurately applying the desired levels of missing data.

Finally, we provide two worked examples of the generation of datasets with variable ploidy: (1) a tetraploid organism, and (2) a diploid organism with two autosomes and a pair of heterogametic sex chromosomes (e.g. similar to a XY human male). In accordance with Wilson Sayres (2018), we expected the following levels of average genetic diversity in the second example: autosomes $4N_e\mu$, X chromosome $3N_e\mu$ (¾ of autosomes), and Y chromosome $N_e\mu$ (¼ of autosomes).

All code used in the validation procedures and example plots for *vcfsim* are available as a Github repository: https://github.com/Pie115/*vcfsim*-benchmarking-and-test-analysis.

Results

Accuracy vs. theoretical expectations

The estimated π values calculated from *vcfsim* VCFs using both pixy and VCFtools and closely matched theoretical expectations (Figure 1A, Figure 1B; 4Neu = 0.0068; VCFtools mean π = 0.00676, 95% CI 0.00669-0.00682; pixy mean π = 0.00668, 95% CI 0.00661 = 0.00677, n = 10000). We also observed a perfect linear relationship ($R^2 = 1.00$, correlation test: $p < 0.00001$, n = 100) between the specified amounts of missing data specified by *vcfsim* and the values for missing data measured using bcftools and/or the raw count of rows in the VCF (Figure 1C, Figure 1D). This indicates that *vcfsim* correctly implements arbitrary levels of missing genotypes and sites.

Runtime

The computational runtime to create 1000 VCFs (10 diploid samples, 100000 sites, with a per nucleotide mutation rate of 1e-9 and an effective population size of 1.7M) was 24890.176 seconds total, or an average of 24.89 seconds per VCF.

Worked examples

To showcase additional features in *vcfsim*, we generated representative diploid VCFs displaying missing data (genotypes and sites, Figure 2A), data from a tetraploid organism (Figure 2B), as well as data from a diploid organism with heterogametic sex chromosomes (Figure 3). When multiple ploidy levels are simulated in this manner, the genetic diversity of autosomes and sex chromosomes are consistent with theoretical expectations (Figure 3, $4N_e\mu$ for autosomes, $3N_e\mu$ for X chromosomes, and $N_e\mu$ for Y chromosomes).

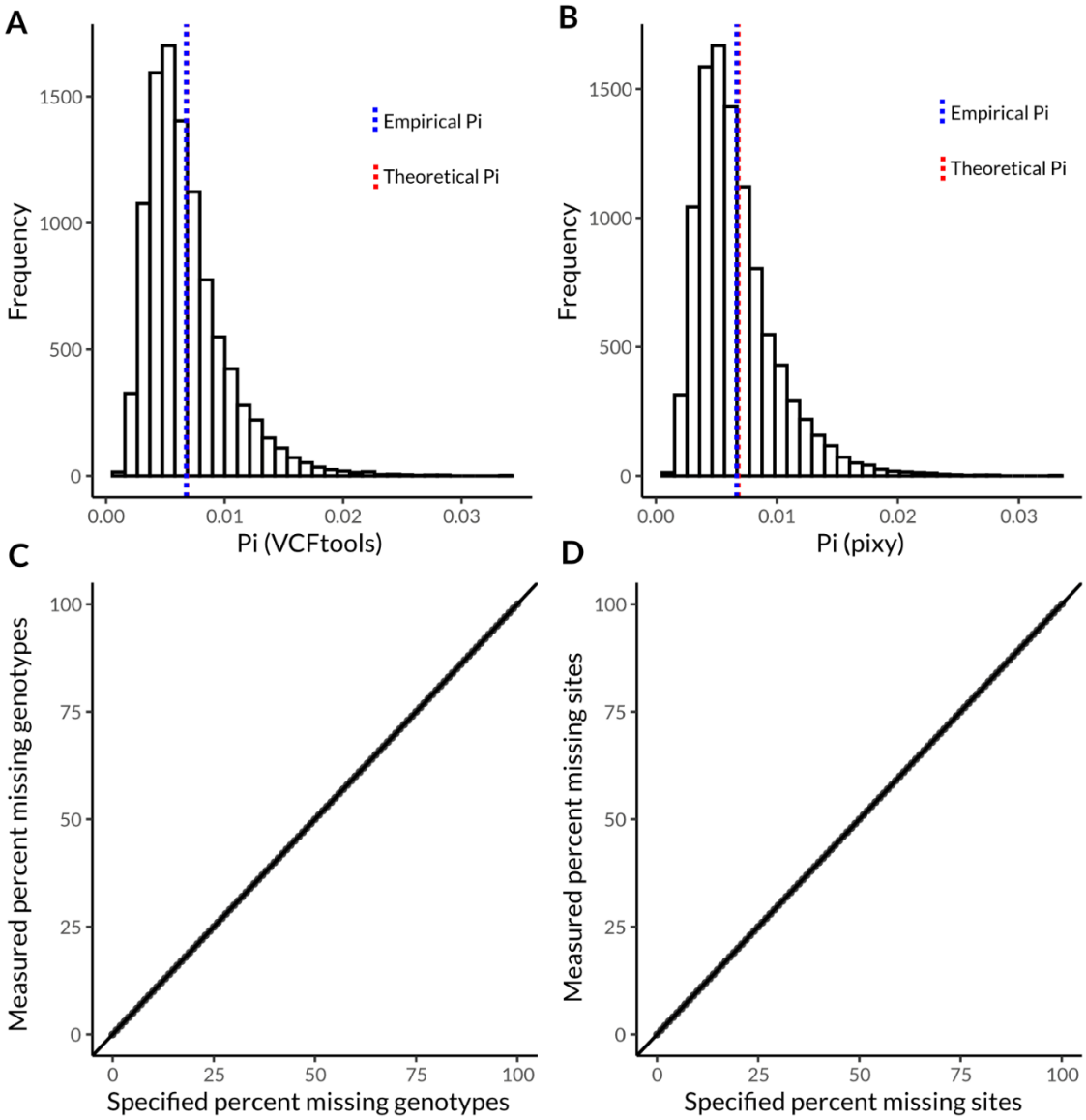

Figure 1 | Distribution and accuracy of nucleotide diversity (pi) values and missing data in VCFs produced by *vcfsim*. (a) Distribution of pi values calculated with VCFtools in the absence of missing data. (b) Distribution of pi values calculated with pixy in the absence of missing data. Vertical lines in (a) and (b) represent the theoretical expectation for pi (red, $4N_e\mu$), and the empirical mean of the computed values (blue). (c) Relationship between specified (*vcfsim*) and measured (bcftools) percentages of missing genotypes. (d) Relationship between specified (*vcfsim*) and measured (manual count) percentages of missing sites.

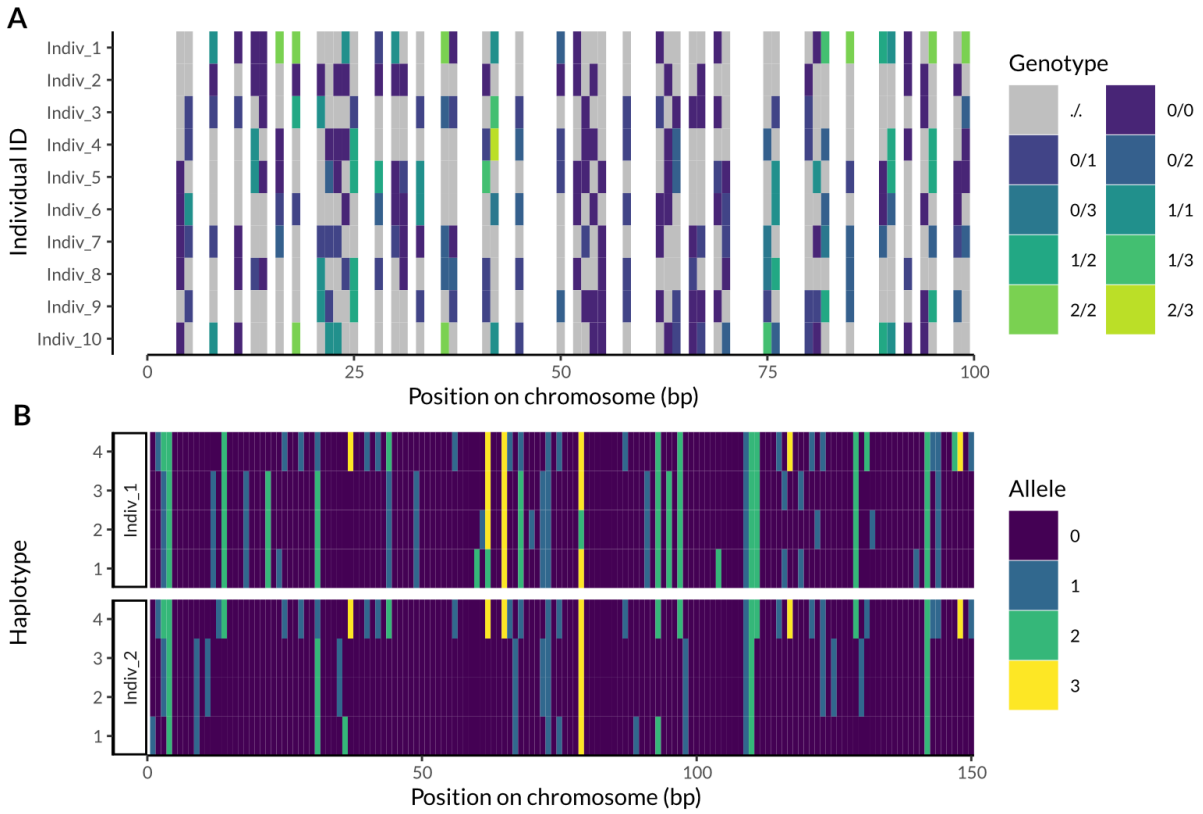

Figure 2 | A heatmap visualization of VCF data generated using *vcfsim*. (A) A 100 bp span of genetic data from ten diploid individuals simulated using *vcfsim*. Each column represents a site in the genome, and each row a diploid individual. Each cell is colored based on its diploid genotype (note some sites are polyallelic), with grey representing a missing genotype. Completely missing sites are represented as white. (B) Simulated VCF of data from two tetraploid individuals with no missing data. Each column represents a site in the genome, and each row one of the individual haplotypes (1-4) for each individual (Indiv_1 and Indiv_2), with cells colored based on the haploid genotype (i.e. allele) at each site. Alleles in both plots are arbitrarily coded 0-4, with zero being the reference allele.

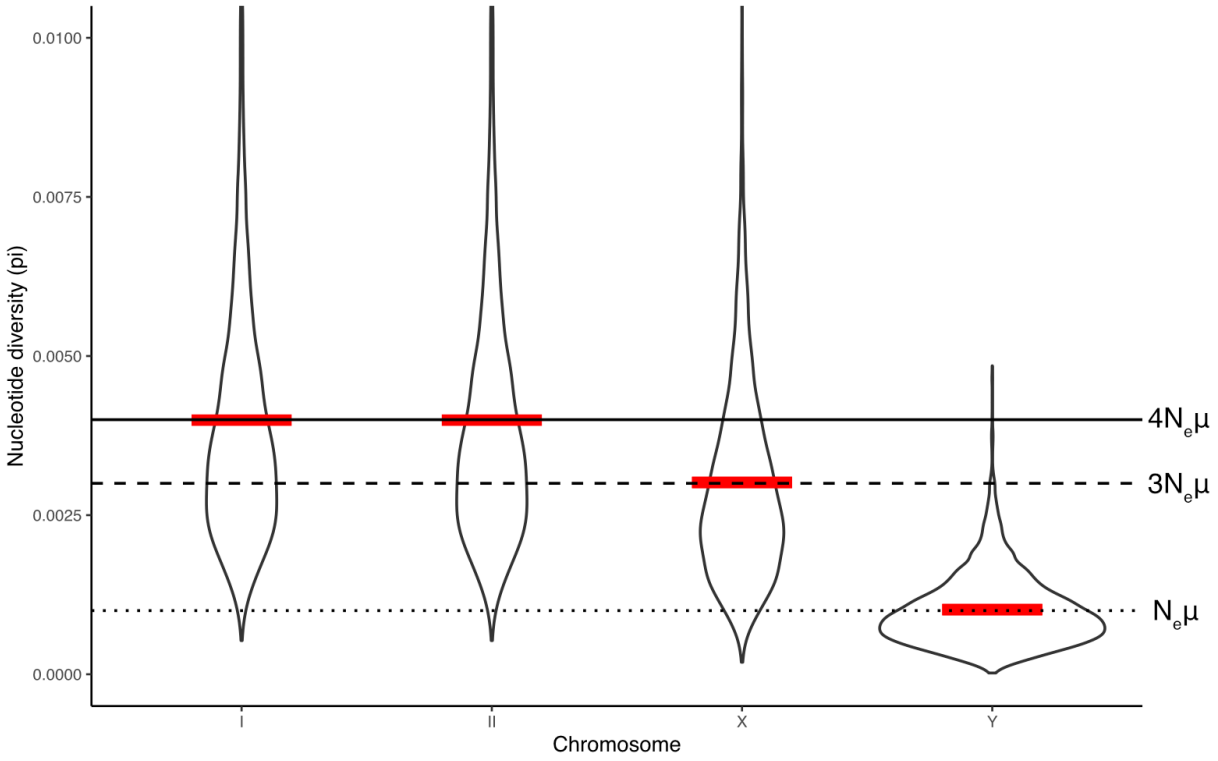

Figure 3 | A violin plot of nucleotide diversity (pi) measured in 10000 bp windows in a diploid individual with autosomes (I, II) and heterogametic sex chromosomes (X, Y). Each violin shows the frequency distribution (density) for nucleotide diversity on each chromosome. Horizontal lines show the theoretical expectation for each class of chromosome (autosomes = $4N_e\mu$, X = 3 $N_e\mu$, Y = $N_e\mu$). Red lines depict the empirical mean nucleotide diversity for the whole chromosome (expected to match the horizontal black lines).

Discussion

vcfsim provides a standardized method for generating all-sites VCFs, complete with configurable levels of missing data and flexible ploidy configurations. It provides a simple and customizable interface to the widely used coalescent simulation msprime (Baumdicker et al., 2022), along with parameterized postprocessing of msprime VCFs to add invariant and missing sites. The ability to customize these parameters allows for the creation of test datasets that closely mimic real-world sequencing outputs, making it a useful tool for a variety of applications in genomics and population genetics.

We demonstrate that the VCFs generated by *vcfsim* align with the theoretical expectations of nucleotide diversity, with no errors introduced by the simulation or post-processing steps (Figure 1A and 1B). Missing data (sites and individual genotypes) are also exactly represented as specified in all output files. Together, these results validate the accuracy of *vcfsim* in replicating expected genetic diversity and missing data in the simulated VCF files.

Our results also demonstrate that *vcfsim* is highly flexible, and capable of generating VCFs with varied ploidy levels and missing data. For example, simulations of tetraploid organisms (Figure 2B) and diploid organisms with heterogametic sex chromosomes (Figure 2C) accurately reflect theoretical diversity expectations. As such, *vcfsim* will be a useful tool across a wide variety of biological systems.

While a relatively straightforward tool, *vcfsim* has a large number of potential applications. One prime use of *vcfsim* is to benchmark software tools designed for genetic analysis, ensuring that they perform correctly across different configurations of missing data or ploidy. Additionally, *vcfsim* could be used to generate datasets for training different machine learning models, such as neural networks aimed at identifying genetic variants or inferring population structure. These datasets could also be used to conduct power analyses for population genetic inference, helping researchers assess how different levels of missing data or sample sizes affect the ability to detect evolutionary signals or demographic events.

Limitations

While *vcfsim* provides an accurate and flexible tool for simulating VCFs, there are several important limitations to consider. First, *vcfsim* simulates VCFs with either completely known or completely missing data. In real-world sequencing data, however, we often encounter intermediate cases, such as sites with low sequencing depth or uncertain genotype likelihoods. *vcfsim* does not currently simulate these complexities, meaning that it does not account for partial data or the uncertainties that can arise from sequencing errors or low coverage. This simplification makes it harder to replicate the full range of real-world sequencing issues, which could limit the usefulness of the simulations in cases where genotype likelihoods or sequencing depth are important for analysis.

Another limitation is the size of the all-sites VCFs generated by *vcfsim*. Because these files include both variant and invariant sites, they can become quite large, particularly for genomes with many invariant regions or when simulating across multiple chromosomes. While this feature is essential for certain types of analyses (Korunes & Samuk, 2021), it can create challenges in terms of storage and computational efficiency. Handling large all-sites VCFs may require significant memory and processing resources, which can be a bottleneck in large-scale simulations or analyses.

Conclusion

vcfsim provides a much-needed tool for generating all-sites VCFs with customizable missing data and ploidy configurations. By addressing several limitations in current VCF generation workflows, *vcfsim*

simplifies the process of creating realistic test datasets for use in genomic research and software validation. The ability to accurately simulate and format genetic variation and introduce controlled amounts of missing data opens up new possibilities for benchmarking tools, training machine learning models, and conducting power analyses in population genetics.

References

Baumdicker, F., Bisschop, G., Goldstein, D., Gower, G., Ragsdale, A. P., Tsambos, G., ... Kelleher, J. (2022). Efficient ancestry and mutation simulation with msprime 1.0. Genetics, 220(3). doi:10.1093/genetics/iyab229

conda contributors. 2025. *Conda: A System-Level, Binary Package and Environment Manager Running on All Major Operating Systems and Platforms*. Zenodo. https://doi.org/10.5281/ZENODO.14680664.

Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., DePristo, M. A., ... 1000 Genomes Project Analysis Group. (2011). The variant call format and VCFtools. Bioinformatics , 27(15), 2156–2158.

Danecek, P., Bonfield, J. K., Liddle, J., Marshall, J., Ohan, V., Pollard, M. O., ... Li, H. (2021). Twelve years of SAMtools and BCFtools. GigaScience, 10(2). doi:10.1093/gigascience/giab008

Grüning, Björn, Ryan Dale, Andreas Sjödin, Brad A. Chapman, Jillian Rowe, Christopher H. Tomkins-Tinch, Renan Valieris, Johannes Köster, and Bioconda Team. 2018. "Bioconda: Sustainable and Comprehensive Software Distribution for the Life Sciences." *Nature Methods* 15 (7): 475–76.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. "Array Programming with NumPy." *Nature* 585 (7825): 357–62.

Korunes, K. L., & Samuk, K. (2021). pixy: Unbiased estimation of nucleotide diversity and divergence in the presence of missing data. Molecular Ecology Resources, 21(4), 1359–1368.

Lowry, D. B. (2017). Breaking RAD: an evaluation of the utility of restriction site-associated DNA sequencing for genome scans of adaptation. Molecular Ecology Resources, 17(2), 142–152.

Samuk, K. (2023). Average nucleotide diversity should be weighted by per-site sample size. Molecular Ecology Resources, 23(2), 355–358.

Perez, Fernando, and Brian E. Granger. 2007. "IPython: A System for Interactive Scientific Computing." *Computing in Science & Engineering* 9 (3): 21–29.

The Pandas development team. (2024). pandas-dev/pandas: Pandas. doi:10.5281/ZENODO.13819579

Wilson Sayres, M. A. (2018). Genetic Diversity on the Sex Chromosomes. Genome Biology and

Evolution, 10(4), 1064–1078.

Wong, K. Y., Zeng, D., & Lin, D. Y. (2019). Robust Score Tests With Missing Data in Genomics Studies. Journal of the American Statistical Association, 114(528), 1778–1786.