SOFTWARE TOOL ARTICLE

REVISED ngsJulia: population genetic analysis of next-generation DNA sequencing data with Julia language [version 3; peer review: 1 approved, 3 approved with reservations]

Alex Mas-Sandoval[1], Chenyu Jin[1,2], Marco Fracassetti[3], Matteo Fumagalli[1,4]

[1]Department of Life Sciences, Imperial College London, London, UK
[2]Institute of population genetics, University of Veterinary Medicine Vienna, Vienna, Austria
[3]Department of Ecology, Environment and Plant Sciences, Stockholm University, Stockholm, Sweden
[4]School of Biological and Behavioural Science, Queen Mary, University of London, London, UK

v3 **First published:** 31 Jan 2022, **11**:126
https://doi.org/10.12688/f1000research.104368.1
Second version: 29 Nov 2022, **11**:126
https://doi.org/10.12688/f1000research.104368.2
Latest published: 14 Jul 2023, **11**:126
https://doi.org/10.12688/f1000research.104368.3

Abstract
A sound analysis of DNA sequencing data is important to extract meaningful information and infer quantities of interest. Sequencing and mapping errors coupled with low and variable coverage hamper the identification of genotypes and variants and the estimation of population genetic parameters. Methods and implementations to estimate population genetic parameters from sequencing data available nowadays either are suitable for the analysis of genomes from model organisms only, require moderate sequencing coverage, or are not easily adaptable to specific applications. To address these issues, we introduce ngsJulia, a collection of templates and functions in Julia language to process short-read sequencing data for population genetic analysis. We further describe two implementations, ngsPool and ngsPloidy, for the analysis of pooled sequencing data and polyploid genomes, respectively. Through simulations, we illustrate the performance of estimating various population genetic parameters using these implementations, using both established and novel statistical methods. These results inform on optimal experimental design and demonstrate the applicability of methods in ngsJulia to estimate parameters of interest even from low coverage sequencing data. ngsJulia provide users with a flexible and efficient framework for ad hoc analysis of sequencing data.ngsJulia is available from: https://github.com/mfumagalli/ngsJulia.

Keywords
high-throughput sequencing data, population genetics, genotype likelihoods, Julia language, pooled sequencing, polyploidy, aneuploidy

Open Peer Review

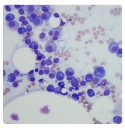
Approval Status ? ✓ ? ?

	1	2	3	4
version 3 (revision) 14 Jul 2023		✓ view	? view	? view
version 2 (revision) 29 Nov 2022		? view		
version 1 31 Jan 2022	? view			

1. **Lindsay Clark**, University of Illinois at Urbana-Champaign, Urbana, USA
2. **Stéphane De Mita**, Institut national de la recherche agronomique (INRAE), Paris, France
3. **Nikolay Oskolkov**, Lund University, Lund, Sweden
4. **Jens Léon**, University of Bonn, Bonn, Germany
Michael Schneider, Forschungsinstitut fur Biologischen Landbau, Frick, Switzerland

Any reports and responses or comments on the article can be found at the end of the article.

This article is included in the RPackage gateway.



This article is included in the Cell & Molecular Biology gateway.

Corresponding author: Matteo Fumagalli (m.fumagalli@qmul.ac.uk)

Author roles: Mas-Sandoval A: Investigation, Visualization, Writing – Review & Editing; **Jin C**: Investigation, Software, Writing – Review & Editing; **Fracassetti M**: Data Curation, Writing – Review & Editing; **Fumagalli M**: Conceptualization, Formal Analysis, Funding Acquisition, Investigation, Methodology, Project Administration, Software, Visualization, Writing – Original Draft Preparation

Competing interests: No competing interests were disclosed.

Grant information: MF and AMS are funded The Leverhulme Research Project Grant RPG-2018-208.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: Mas-Sandoval A, Jin C, Fracassetti M and Fumagalli M. **ngsJulia: population genetic analysis of next-generation DNA sequencing data with Julia language [version 3; peer review: 1 approved, 3 approved with reservations]** F1000Research 2023, **11**:126 https://doi.org/10.12688/f1000research.104368.3

First published: 31 Jan 2022, **11**:126 https://doi.org/10.12688/f1000research.104368.1

REVISED **Amendments from Version 2**

In the revised version, we provide users with a comprehensive online documentation of APIs and CLIs at https://ngsjulia.readthedocs.io. Additionally, all functions have docstring documentation. The methodology behind some of the implementations has been clarified and references on the applicability of said methods have been added. We also rephrased few sentences for clarity as suggested by the reviewer and fixed few typos. We believe that these changes significantly enhance the user's experience and simplify the interpretation of results obtained using `ngsJulia`.

Any further responses from the reviewers can be found at the end of the article

Introduction

Population genetics, i.e., the study of genetic variation within and between groups, plays a central role in evolutionary inferences. The quantification of genetic diversity serves the basis for the inference of neutral[1] and adaptive[2] events that characterised the history of different populations. Additionally, the comparison of allele frequencies between groups (i.e. cases and controls) is an important aspect in biomedical and clinical sciences.[3]

In the last 20 years, next-generation sequencing (NGS) technologies allowed researchers to generate unprecedented amount of genomic data for a wide range of organisms.[4] This revolution transformed population genetics (therefore also labelled as population genomics) to a data-driven discipline. Data produced by short-read sequencing machines (still the most accessible platform worldwide) consist of a collection of relatively short (approx. 100 base pairs) fragments of DNA which are then mapped or *de novo* assembled to form a contiguous sequence.[4] At each genomic position, all observed sequenced reads are used to infer the per-sample genotype (an operation called 'genotype calling') and the inter-samples variability, i.e., whether a particular site is polymorphic (an operation called 'single-nucleotide polymorphism (SNP) calling').[5]

To this end, several software packages have been implemented to perform genotype and SNP calling from NGS data, the most popular ones being `samtools/bcftools`,[6] `GATK`,[7] and `freeBayes`.[8] When, on average, few reads map at each genomic position (a scenario referred to as 'low-coverage' or 'low-depth'), genotypes and SNPs cannot be assigned with confidence due to the high data uncertainty.[9,10] Under these circumstances, statistical methods that integrate data uncertainty into genotype likelihoods and propagate it to downstream analyses have been proposed.[5] Software packages like `ANGSD`[11] and `ngsTools`,[12] among others reviewed by Lou *et al.*,[13] implement a statistical framework to estimate population genetic metrics from low-coverage sequencing data. Similarly, an affordable generation of sequencing data from large sample sizes can be obtained via pooled sequencing experiments, where assignment of individual samples is typically not retained.[14] Several new and popular software for the analysis of pooled sequencing have been proposed in recent years.[15,16]

Despite these advances, most of these implementations are either tuned and suitable for model organisms only (e.g., with haploid or diploid genomes) or not easily adaptable to novel applications. Therefore, an accessible computational framework for building and testing *ad hoc* population genetic analyses from NGS data is in dire need. Among programming languages, `Julia`[17] has emerged as both a powerful and easy-to-use dynamically typed language that is widely used in many fields of data sciences, including genomics.[18] While several `Julia` packages are currently available for both population genetic and bioinformatic analyses (e.g., BioJulia), to our knowledge, a suitable framework for custom population genetic analysis from NGS data is not available yet.

Here we present `ngsJulia`, a set of templates and functions in `Julia` language to process NGS data and create custom analyses in population genetics. To illustrate its applicability, we further introduce two implementations, `ngsPool` and `ngsPloidy`, for the analysis of pooled sequencing data and polyploid genomes, respectively. By extensive simulations, we show the performance of several methods implemented in these programs under various experimental conditions. We also introduce novel statistical methods to estimate population genetic parameters from NGS data and demonstrate their applicability and suggest optimal experimental design. We finally discuss further directions and purposes for `ngsJulia` and bioinformatics for NGS data analysis.

Methods

Implementation

`ngsJulia` was built in `Julia` language (Julia Programming Language, RRID:SCR_021666) and requires the packages 'GZip', 'Combinatorics', and 'ArgParse'. Auxiliary scripts to process output files were built in `R` (`R` Project for Statistical Computing, RRID:SCR_001905) version 3.6.3 and require the package 'getopt'. `ngsJulia` receives

gzipped mpileup input files which can be generated using `samtools`.[19] Output files are in text file format and can be easily parsed for producing summary plots and for further analyses. Scripts for some downstream analyses are provided in `ngsJulia`.

Operation

`ngsJulia` is compatible with all major operating systems and is maintained at https://github.com/mfumagalli/ngsJulia. Documentation and tutorials are available via this GitHub repository and archived at Zenodo[20] at the time of writing. All analyses in the manuscript are performed in `Julia` language and `R`.

`ngsJulia` implements functions to read and parse gzipped mpileup files and to output gzipped text files on various calculations (e.g., genotype and allele frequency likelihoods) and estimations (e.g., allele frequencies), as requested by the user. `ngsJulia` also allows for several data filtering options, including on global and per-sample depth, proportion or count of minor allele, and base quality. Finally, several options for SNP and biallelic and triallelic polymorphisms calling are available.

Nucleotide, genotype and allele frequency likelihoods

`ngsJulia` provides utilities to calculate nucleotide and genotype likelihoods, i.e., the probability of observed sequencing data given a specific nucleotide or genotype,[21] for an arbitrary ploidy level, as in Soraggi *et al.*[22] We now describe how such quantities are calculated in `ngsJulia`.

Following the notation in Soraggi *et al.*,[22] for one sample and one site, we let O be the observed NGS data, Y the ploidy, and G the genotype. Therefore, G has values in $\{0,1,\ldots,Y\}$, i.e., the number of derived (or alternate) alleles.

In the simplest form, genotype likelihoods can be calculated by considering individual base qualities as probabilities of observing an incorrect nucleotide.[21] We adopt the calculation of genotype likelihoods for an arbitrary ploidy level $P(O|G,Y)$ as proposed in Soraggi *et al.*[22] From the genotype likelihoods with $P(O|G,Y=1)$, the two most likely alleles are identified by sorting $P(O|G,Y=1)$ values after pooling all sequencing reads together across all samples. This operation will restrict the range of possible genotypes to biallelic variation only. Note that this calculation is still valid for monomorphic sites, although the actual assignment of the minor allele is meaningless.

We now describe how to estimate $F_{a,n}$, the frequency of allele $a \in \{A,C,G,T\}$ at site n. Similarly to Kim *et al.*,[23] the log-likelihood function for $F_{a,n}$ is given by:

$$\log\left(P(O_n|F_{a,n})\right) = \sum_{i=1}^{C_n} P(O_n|c_{i,n}=a, Y=1)F_n \tag{1}$$

where $c_{i,n}$ is the $i-th$ read at site n, and C_n is the total depth across all samples. Function 1 is maximised to obtain a maximum likelihood estimate (MLE) of the sample allele frequency, $\widehat{F}_n$, either with a grid- or golden-section- search algorithm.

SNP calling

To perform SNP calling, we implement a likelihood-ratio test (LRT) with one degree of freedom with null hypothesis $H_0: F_n = 0$ and alternate hypothesis $H_1: F_n = \widehat{F}_n$, as described by Kim *et al.*[23] Additionally, we develop a test for a site being biallelic or triallelic. The former can be interpreted as a further evidence of polymorphism, while the latter as a condition not to be met for the site being included in further estimations, as our models assume at most two alleles. The log-likelihood of site n being biallelic is equal to $P(O_n|G_n=\{i,j\}, Y=1)$ while the log-likelihood being triallelic is equal to $P(O_n|G_n=\{i,j,z\}, Y=1)$, with i, j, and z being the most, second most, and third most likely allele with $Y=1$ (i.e. haploid genotype G), respectively. An LRT with one degree of freedom can be conducted to assess whether $P(O_n|G_n=i, Y=1)$ is significantly greater than $P(O_n|G_n=\{i,j\}, Y=1)$, or the latter is significantly greater than $P(O_n|G_n=\{i,j,z\}, Y=1)$.

Allele frequency, site frequency spectrum and association test from pooled sequencing data

Several estimators of population parameters from pooled sequencing data are implemented in `ngsPool`, a separate program which uses functions in `ngsJulia`. We now describe the statistical framework for the analysis of pooled sequencing data.

In case of data with unknown sample size, the MLE of the population allele frequency F_n is calculated as in Equation 1 with $F \in (0,1)$. With known sample size, the same equation is used to calculate sample allele frequency likelihoods $P(O_n|F_n = f)$,[24] for instance with $f \in \{0,1,\ldots,M \times Y\}$ for M samples of equal ploidy Y. From these likelihoods, we can calculate both the MLE and the expected value with uniform prior probability as estimators of F_n.

A simple estimator of the site frequency spectrum (SFS) from pooled sequencing data is obtained by counting point-estimates of F_n across all sites. We propose a novel estimator of the SFS implemented in `ngsPool`. Under the standard coalescent model with infinite sites mutations, we let the probability of derived allele frequency F in a sample of N genomes $P(F = f)$ to be proportional to $1/f^K$ with $f \in \{1,\ldots,N-1\}$.[25] The parameter K determines whether the population is deviating from a model of constant effective population size. For instance, $K = 1$ is equal to the expected distribution of $P(F)$ under constant population size, while $K > 1$ models a population shrinking and $K < 1$ population growth.

We optimise the value of K to minimise the Kullback-Liebrel divergence between the expected distribution of $P(F|K)$ and the observed SFS. The latter can be obtained by either counting $\widehat{F}_n$ across all sites or by integrating over the sample allele frequency probabilities $P(F_n = f|O_n) \propto P(O_n|F_n = f)$ (*i.e.* with a uniform prior distribution). A threshold can be set to ignore allele frequencies with low probability to improve computing efficiency and reduce noise. Within this framework, folding spectra can be generated in case of unknown allelic polarisation.

Finally, we introduce a strategy to perform association tests from pooled sequencing data. Similarly to Kim *et al.*,[23] we propose an LRT with one degree of freedom for null hypothesis $H_0 : f_{cases} = f_{controls}$ and alternate hypothesis $H_1 : f_{cases} \neq f_{controls}$. The likelihood of each hypothesis is calculated from $P(O_n|F_n = f)$ and, therefore, this strategy avoids the assignment of counts or per-site allele frequencies. A statistically significant LRT with one degree of freedom suggests a difference in allele frequencies between cases and controls, and possible association between the tested phenotype and alleles.

Ploidy levels and test for multiploidy

We now describe the statistical framework implemented in the program `ngsPloidy` to estimate ploidy levels and test for multiploidy. When multiple samples are available, two scenarios can be envisaged: (i) all samples have the same ploidy, (ii) each sample can have a different ploidy (multiploidy, i.e. as in tumor genomes).[26]

The log-likelihood function for a vector of ploidy levels $\vec{Y}_M = \{Y_1 = y_1, Y_2 = y_2, \ldots, Y_M = y_M\}$ for M samples and N sites is defined as:

$$\log\left(P\left(O|Y = \vec{Y}_M\right)\right) = \sum_{m=1}^{M}\sum_{n=1}^{N} \log\left(\sum_{i \in \{0,1,\ldots,Y_m\}} P(O_n|G_{m,n} = i, Y_m = y_m) P\left(G_{m,n} = i|Y_m = y_m, F_n = \widehat{F}_n\right)\right) \quad (2)$$

With $\widehat{F}_n$ being the MLE of allele frequency at site n. $P(O_n|G_{m,n} = i, Y_m = y_m)$ is the genotype likelihood while $P\left(G_{m,n} = i|Y_m = y_m, F_n = \widehat{F}_n\right)$ is the genotype probability given the ploidy and allele frequency at site n.

Once $\widehat{F}_n$ is estimated and the inbreeding is known (or under the assumption of Hardy-Weinberg Equilibrium (HWE)), then genotype probabilities are fully defined. Equation 2 is optimised by maximising the marginal likelihood of each sample separately, assuming independence among samples and sites.

With limited sample size, $\widehat{F}_n$ is not a good estimator of the population allele frequency and therefore genotype probabilities may not be well defined. In the simplest scenario, genotype probabilities can be set as uniformly distributed, with all genotypes being equally probable. However, the assignment of alleles into ancestral (e.g., wild-type) and derived (e.g., mutant) states is particularly useful to inform on genotype probabilities. Recalling Equation 2, we can substitute $P\left(G_{m,n} = i|Y_m = y_m, F_n = \widehat{F}_n\right)$ with $P(G_{m,n} = i|Y_m = y_m, F_n = \mathbf{E}[F|K]))$, where $\mathbf{E}[F|K]$ is the expected allele frequency of $P(F = f|K) \propto 1/f^K$, as introduced previously. Note that $\mathbf{E}[F|K]$ does not depend on the sequencing data for each specific site n.

Note that, in practice, Equation 2 is a composite likelihood function, as samples and sites are not independent observations due to shared population history and linkage disequilibrium, respectively. A solution to circumvent this issue is to perform a bootstrapping procedure, by sampling with replacement segments of the chromosome and estimate ploidy for each bootstrapped chromosome. The distribution of inferred ploidy levels from bootstrapped chromosomes provides a quantitative measurement of confidence in determining the chromosomal ploidy. It is not possible to calculate

the likelihood of ploidy equal to one after SNP calling. Nevertheless, the identification of haploid genomes from sequencing data is typically trivial.

Unknown or uncertain ancestral allelic state

So far, we assumed to know which allele can be assigned to an ancestral state, and which one to a derived state. However, in some cases, such assignment is either not possible or associated with a certain level of uncertainty due to, for instance, ancestral polymorphisms or outgroup sequence genome from a closely related species not being available. Under these circumstances, we extend our formulation by adding a parameter underlying the probability that the assigned ancestral state is incorrectly identified.

Let us define R as the ancestral state and a as any possible allele in $\{A,C,G,T\}$. In practice, a can take only two possible values as we select only the two most likely alleles. We label this set of the two most common alleles as A and we assume that the true ancestral state is included in such set. The log-likelihood function of ploidy for a single sample m under unknown ancestral state is:

$$\log\left(P(O|Y_M=y_m)=\sum_{n=1}^{N}\log\left(\sum_{a\in A}\sum_{i\in\{0,1,\ldots,Y_m\}}P(O_n|G_{m,n}=i,Y_m=y_m)P\left(G_{m,n}=i|Y_m=y_m,R=a,F_n=\widehat{F}_n\right)P(R=a)\right)\right) \tag{3}$$

Where $P(R=a)$ indicates the probability that allele a is the ancestral state, and it is invariant across sites. If $P(R=a)=0.5$, then the equation refers to the scenario of folded allele frequencies, where each allele is equally probable to be the ancestral state.

Finally, note that in a sufficiently large sample size, the major allele is more probable to represent the ancestral state. This probability depends on the shape of the site frequency spectrum, and it is equal to the cumulative distribution of $P(F|K)$ evaluated at $F=N/2$. We can extend Equation 3 to reflect this parameter uncertainty with a being the major allele in $P(R=a)$.

Test for multiploidy

We introduce a novel test for multiploidy. If all samples have the same ploidy $y\in Y$, then $Y_i=Y_j=y$ is true for all $(i,j)\in 1,2,\ldots,\mathrm{M}$. We propose an LRT for multiploidy with null hypothesis $H_0:\sup\vec{Y}=\{y_1=y,y_2=y,\ldots,y_M=y\}$ and alternate hypothesis $H_1:\vec{Y}=\vec{Y}_{MLE}$ A large value of LRT is suggestive of multiploidy. Statistical significance can be assessed with the LRT and $(M-1)$ degrees of freedom.

Data simulation

To benchmark the performance of methods implemented in `ngsJulia`, we simulated NGS data following a strategy previously proposed by Fumagalli *et al.*[27] available as a stand-alone R script. Briefly, individual genotypes are drawn according to probabilities depending on input parameters. The number of mapped reads at each position is modelled with a Poisson distribution and sequenced bases are sampled with replacement with a probability given by the quality score. As an illustration, the following code

```
Rscript simulMpileup. R --out test.txt --copy 2x10 --sites 1000 \\
--depth 20 --qual 20 ---pool|gzip > test.mpileup.gz
```

will simulate 10 diploid genomes (`--copy 2x10`), 1000 base pairs each (`--sites 1000`) with an average sequencing depth of 20 and base quality of 20 in *Phred* score (`--depth 20 --qual 20`) from pooled sequencing (`--pool`) with results stored in `test.mpileup.gz file`.

For the analysis of pooled sequencing data, we simulated 100,000 independent sites at sample sizes 20, 50, and 100 from a diploid population with a constant effective population size of 10,000. We imposed the average per-sample sequencing depth to be 0.5, 1, 2, or 5 with an average base quality of 20 in *Phred* score. To assess the performance of `ngsPool`, we calculated bias and root mean squared error (RMSE) between the estimated value of the true value, either from the sample or the whole population. While both metrics measure the distance with the true value, the bias retains the direction of the error (i.e. over- or under-estimation). To quantify the accuracy of SNP calling, we calculated F1 scores (the harmonic mean of precision and recall rates).

To simulate data for association test, we assumed an equal number of cases and controls (equal to 150) and 200 SNPs, either causal or non causal. For non causal sites, cases and controls have the same population allele frequency of 0.10. For causal sites, cases and controls have a population allele frequency of 0.09 and 0.04, respectively. These conditions are derived assuming a high risk allele frequency of 0.1, prevalence of 0.2, genotypic relative risk for the heterozygote of 2, genotypic relative risk for the homozygous state of 4. The sample size simulated guarantees at least 80% power with a false positive rate of 0.10.

To illustrate the usage of `ngsPloidy`, we simulated NGS data of genomes with different ploidy levels (one haploid, eight triploids, one tetraploid) at 1000 sites. NGS data was simulated assuming an average depth of 10 at the haploid level. Code and simulated data sets analysed are available in `ngsJulia` GitHub repository.

Empirical data

To showcase the applicability of `ngsJulia` to real data, we deployed to two distinct empirical data sests. To test `ngsPool`, we used mapped reads of *Arabidopsis lyrata* produced in a previous study.[28] Specifically, We used data belonging to the population B where 25 individuals of A. *lyrata* have been sequenced by both pool-sequencing and genotyping-by-sequencing. Since GBS harbours data on a fraction of the genome and Pool-seq is whole genome, We restricted the analysis to the base pairs (bp) sequenced by both techniques, for a total of 15202 bp. The mpileup files have been generated with samtools[6] (v1.14) and SNP calling in GBS data has been performed with Varscan[29] (v2.4.2), resulting in 333 SNPs.

To test `ngsPloidy`, we reanalysed whole-genome sequencing data of the human fungal pathogen *Candida auris* from a previous study.[30] We processed data for 21 samples, each one consisting of 17 contigs. As the null hypotheses is that these samples are haploid, we tested ploidy levels from 1 to 4. We filtered out sites with a minimum depth lower than 5 and a proportion of minor alleles lower than 0.15. The number of sites analysed per contig ranged from 555 to 754.

Results

`ngsJulia` implements data structures and functions for an easy calculation of nucleotide and genotype likelihoods (of arbitrary ploidy) which serve the basis of genotype and SNP calling and for the estimation of allele frequencies and other summary statistics. It is particularly suitable for low-coverage sequencing data and for cases when there is high data uncertainty. To demonstrate the use of `ngsJulia`, we provide two custom applications from its templates and functions.

ngsPool: analysis of pooled sequencing data

We used `ngsJulia` to implement a separate program, called `ngsPool`, to perform population genetic analysis from pooled sequencing data. Specifically, `ngsPool` implements established and novel statistical methods to estimate allele frequencies and site frequency spectra (SFS) and perform association tests from pooled sequencing data.

`ngsPool` uses functions in `ngsJulia` to parse mpileup files as input. As an illustration, the following code

```
julia ngsPool.jl --fin test.mpileup.gz --fout test.out.gz --lrtSnp 6.64
```

will parse `test.mpileup.gz` file and write estimates of allele frequencies in `test.out.gz` file from unknwon sample size after performing SNP calling with an LRT value of 6.64 (`--lrtSnp 6.64`, equivalent to a p-value of 0.01).

Depending on the options selected by the user, output files contain, various results are printed on the screen, including

- inferred major allele,
- inferred minor allele,
- LRT statistic for SNP calling,
- LRT for bi- and tri-allelic calling,
- three estimators of the minor allele frequency at each site.

Additionally, `ngsPool` can output a file with per-site sample allele frequency likelihoods. The following code

```
julia ngsPool.jl --fin test.mpileup.gz --fout test.out.gz \\
--nChroms 20 --fsaf test.saf.gz
```

will produce estimates of allele frequencies from the known sample size (specified by `--nChroms 20`) and allele frequency likelihoods in `test.saf.gz file`.

These files can then be used to estimate the SFS and perform an association test using two scripts provided in `ngsPool`. For instance, the code

```
Rscript poolSFS. R test.saf.gz > sfs.txt
```

will estimate the SFS, while the code

```
Rscript poolAssoc. R test.cases.saf.gz test.controls.saf.gz > assoc.txt
```

will perform an association test assuming two sets of allele frequency likelihood files, one from cases (`test.cases.saf.gz`) and one from controls (`test.controls.saf.gz`).

Estimation of allele frequencies

To illustrate the usage of `ngsPool`, we estimated allele frequencies based on simulated data at different experimental conditions. We sought to compare the performance among different estimators implemented in the program. If the sample size is not provided, `ngsPool` provides a simple MLE of the population allele frequency assuming haploidy. Alternatively, if the sample size is provided by the user, `ngsPool` calculates sample allele frequency likelihoods and returns both the MLE and the expected value of the allele frequency using a uniform prior probability.

Results show that the error of estimating allele frequencies decreases with increasing depth and sample size, as expected (Figure 1). Likewise, the error for estimating the population allele frequency is more pronounced for lower sample sizes. MLE values tend to be less biased than expected values and sample estimates appear to be unbiased even at depth 1 for moderate sample size (Figure 1).

Furthermore, we simulated NGS data at fixed population allele frequency and compared the distribution of true sample allele frequencies and estimated values using a MLE approach from known sample size. Figure 2 shows that most of the deviation from the true population allele frequencies occurs at intermediate frequencies (*F* equal to 0.5). This effect is more evident for low depth and low sample size (Figure 2).

We then assessed the effect of low-frequency variants on SNP calling. Specifically, we calculated F1 scores for SNP calling when the population allele frequency is 0 (not a SNP) or greater than 0 (a SNP). Figure 3 shows how the prediction

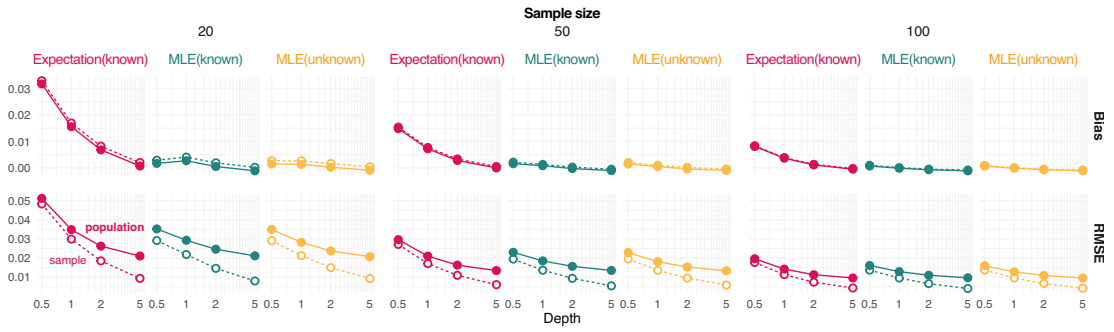


Figure 1. Estimation of minor allele frequencies from pooled sequencing data. Bias and root mean square error (RMSE) against the reference true value from either the population or sample are provided for each value of depth *D* (the average number of sequenced reads per base pair) and sample size (on column panels) tested. Three estimators of allele frequencies are considered: a population maximum likelihood estimate (MLE) from unknown sample size and the expected value and MLE from known sample size.

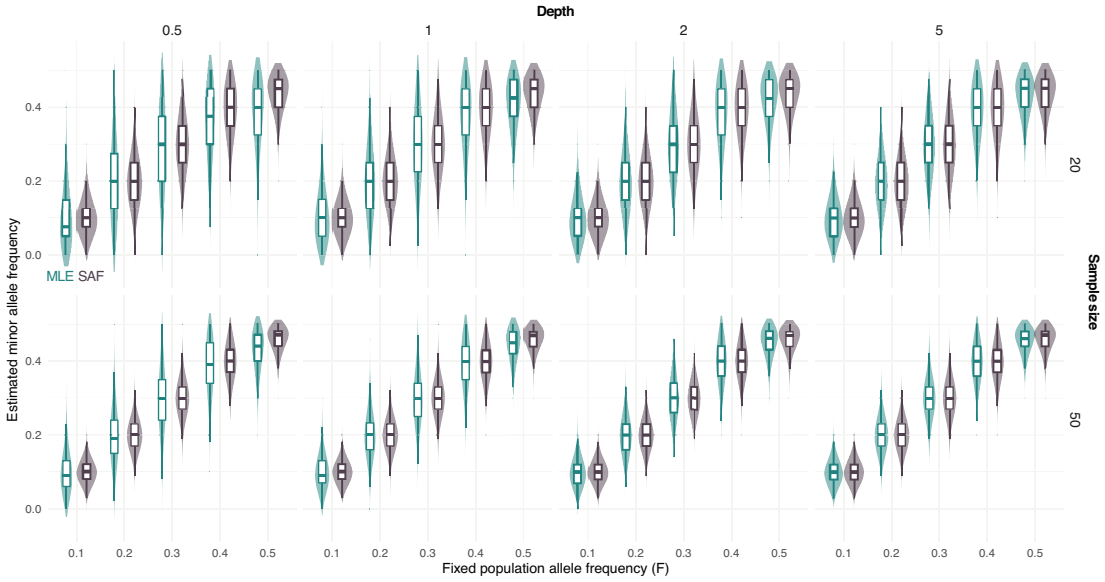


Figure 2. Distribution of estimated minor allele frequencies from pooled sequencing data. True sample allele frequencies (SAF) and maximum likelihood estimates (MLE) are shown at different fixed population allele frequencies *F* and sample sizes (on rows, 20 and 50) and depths (the average number of sequenced reads per base pair, on columns, 0.5, 1, 2, and 5).

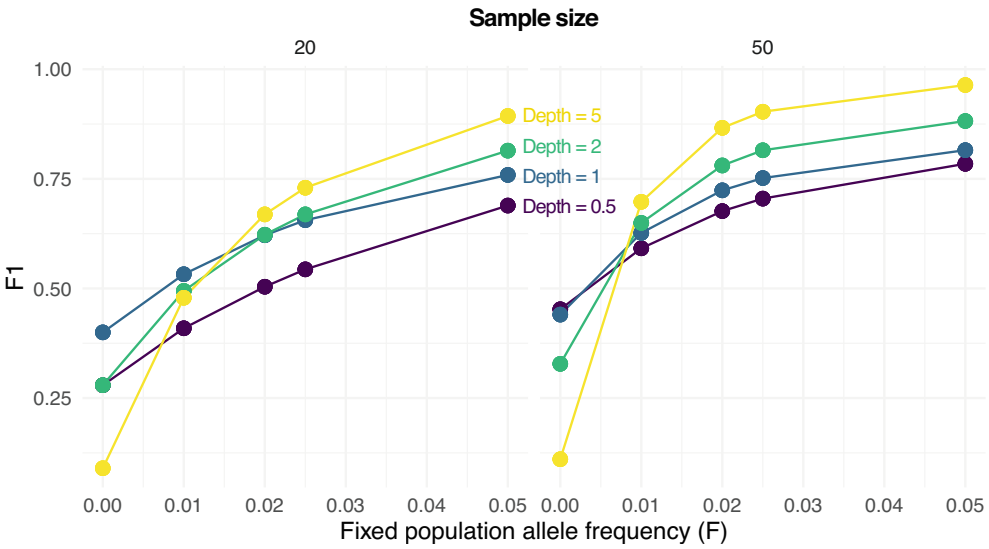


Figure 3. Performance of single nucleotide polymorphism (SNP) calling from pooled sequencing data. F1 scores (harmonic means of precision and recall rates) for predicting either a SNP (true population allele frequency *F* greater than 0) or not ($F = 0$) are reported are various values of *F*, sample sizes (on columns, 20 and 50) and depths *D* (the average number of sequenced reads per base pair).

performance increases with the population allele frequency (F), sample size, and depth (D). For instance, an F1 score greater than 0.75 is obtained with 20 samples only for $F = 0.05$ and $D > 0.5$. On the other hand, the same F1 score is achieved with 50 samples even with $F = 0.025$ and also at $F = 0.02$ but only if $D > 1$.

SNP calling under-performs when there is no variation in the population ($F = 0$), with sequencing errors and sampling statistical uncertainty generating estimates of F greater than 0.

`ngsPool` implements several methods to estimate the SFS from sample allele frequencies. As described in the methods, a simpler estimator is based on assigning the most likely sample allele frequency at each site (labelled count). `ngsPool` implements novel estimators of SFS from pooled sequencing data as described in the method section. A script implements an algorithm to fit the theoretical SFS to the observed SFS. The latter can be calculated either by assigning per-site MLE of allele frequencies (labelled `Fit_count`) or by integrating the uncertainties across all sample allele frequency likelihoods (labelled `Fit_afl`).

Figure 4 shows the error in estimating either the population or sample SFS at various settings with different methods. The error decreases with increasing depth and sample size. Estimating SFS by fitting the theoretical SFS without assignment of allele frequencies generally outperforms other tested strategies (Figure 4).

Notably, the novel approach implemented in `ngsPool` to estimate the parameter K of the SFS distribution (see Methods) allow us to directly quantify the error in inferring demographic events. In fact, all simulations assumed constant population size, equivalent to $K = 1$. Figure 5 shows the estimated values of K by fitting the SFS either by assigning (counting) allele frequencies (`Fit_count`) or by using allele frequency likelihoods (`Fit_afl`). For low-to-moderate depth and sample size, estimates of K tend to suggest population expansion ($K < 1$), possibly due to an over-estimation of the abundance of low-frequency alleles. However, the error is reduced when integrating the data uncertainty with sample allele frequency likelihoods, as estimates of K values tend to be closer to the true simulated value of 1 (Figure 5).

`ngsPool` implements a script to perform association tests from pooled sequencing data. Specifically, the script calculates an LRT statistic, with null hypothesis being that allele frequencies of cases and controls (or any two groups) are the same, as used by Kim *et al.*[23] It uses sample allele frequency likelihoods and, therefore, it maintains data uncertainty and avoids the assignment of counts or per-site allele frequencies. An LRT statistics significantly greater than 0 indicates a difference in allele frequencies between cases and controls.

Figure 6 compares the distribution of LRT statistics between causal and non causal sites at different experimental scenarios. The distribution of LRT at causal SNPs is skewed towards higher values for increasing depth, indicating more support to find phenptype-SNP association. Nevertheless, a clear separation between the distributions of causal and non causal SNPs is observed at low depth (Figure 6).

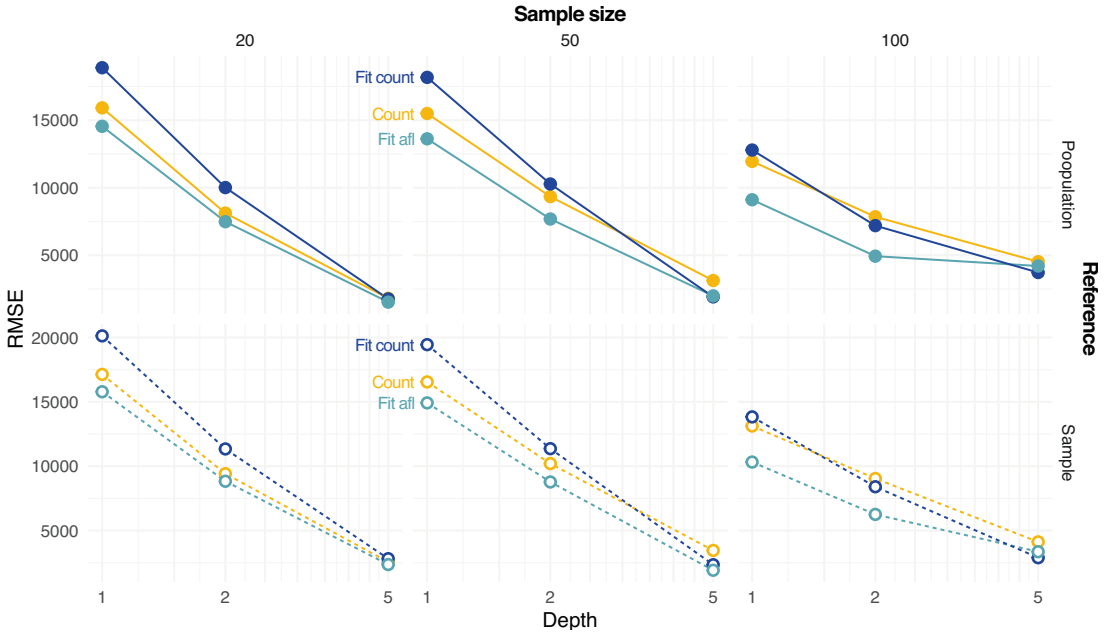


Figure 4. Estimation of site frequency spectrum (SFS) from pooled sequencing data. Estimates based on counting allele frequencies (Count) or fitting from counted allele frequencies (`Fit Count`) or from allele frequency likelihoods (`Fit afl`) are calculated. Root mean square error (RMSE) values between true (either population or sample, on rows) and estimated SFS are reported at various depths *D* and sample sizes (on columns).

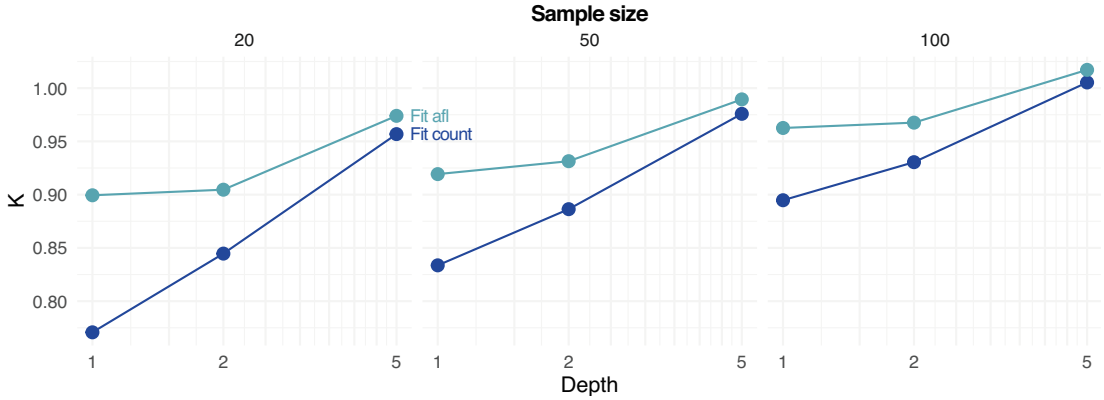


Figure 5. Estimation of parameter *K* (used to determine if the population is deviating from constant effective population size) of the site frequency spectrum at different depths D and sample sizes (on rows) from pooled sequencing data. Estimates based on fitting from counted allele frequencies (`Fit count`) and from allele frequency likelihoods (`Fit afl`) are reported. Note that the true simulated value of *K* is 1.

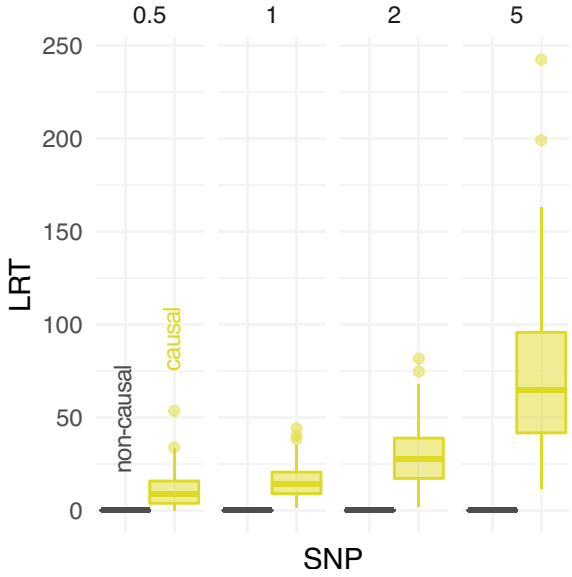


Figure 6. Performance of association test from pooled sequencing data. The distribution of likelihood-ratio test (LRT) statistics for casual and non causal single-nucleotide polymorphism (SNP) is reported at different depths (the average number of sequenced reads per base pair, on columns).

To illustrate the application of ngsPool to real data, we reanalysed genomic data from *Arabidopsis lyrata*.[28] In this study, authors generated data both by genotype-by-sequencing and by pooled-sequencing, with the former providing ground-truth values for genotype and allele calls. Estimates of minor allele frequencies using `ngsPool` yield a lower RMSE (0.09372) than estimates from Varscan (0.09783) across all SNPs analysed.

ngsPloidy: analysis of sequencing data from polyploid genomes

We further utilised `ngsJulia` to implement an additional program, `ngsPloidy`, for the estimation of ploidy from unknown genotypes. The method implemented is similar to the one proposed by Soraggi *et al.*[22] with some notable differences on the calculation of genotype probabilities (see Methods). Additionally, `ngsPloidy` includes a novel method to test for multiploidy in the sample. As an illustration, the following code

```
julia ngsPloidy.jl --fin test.mpileup.gz --nSamples 20 > test.out
Rscript ploidyLRT. R test.out
```

will estimate ploidy levels for 20 genomes (`--nSamples 20`) from `test.mpileup.gz` file and return LRT values.

Following Equation 2, the genotype probabilities for each tested ploidy are pre-calculated using a script provided in `ngsPloidy`. This script takes as input the value of parameter *K* (the shape of the expected SFS), the effective population size, and the probability that the major allele is the ancestral allele. The latter can be either set by the user (e.g., a value of 0.5 would be equivalent to unknown polarisation, as in a folded SFS) or be calculated from the expected population SFS itself. If genotype probabilities are not set, a uniform distribution is assigned. Further options allow for estimation only on called SNPs and/or genotypes.

`ngsPloidy` uses functions in `ngsJulia` to parse mpileup files as input. At the end of the computation, various results are printed on the screen, including

- the number of analysed sites that passed filtering for each sample,
- a matrix of ploidy log-likelihoods for each sample,
- the log-likelihood and MLE of the ploidy vector (i.e., the individually estimated ploidy for each sample),
- LRT scores for the test of multiploidy against all tested ploidy levels.

Additionally, if requested by the user, `ngsPloidy` can generate output files with several statistics for each site (e.g., estimate allele frequency), and all per-site genotype likelihoods for each sample and the tested ploidy.

To illustrate the usage of `ngsPloidy`, we deployed it to simulated data of an multiploid sample consisting of one diploid, eight triploid and one tetraploid genomes. We compared the performance of ploidy and multiploidy inference among different choices of genotype probabilities. The latter were derived either from the expected folded population site frequency, from the estimated ancestral population allele frequency, or from the calculated sample allele frequency.

In all tested cases, we inferred the correct vector of marginal ploidy levels. We therefore assessed the confidence in such inference by calculating the LRT statistics of ploidy and multiploidy inferences. Both were calculated by comparing the most against the second most likely vector of ploidy levels or the most likely vector of equal ploidy levels, respectively.

Results show that using the per-site estimate sampled allele frequency yields higher LRT statistics (and therefore confidence) than using expected population allele frequencies (Table 1). We reiterate that for all three cases we correctly identified the patterns of multiploidy. However, we should caution that with lower sample sizes we do not expect inferences using estimated sample allele frequencies to perform well.

To illustrate the applicability of `ngsPloidy` to real empirical data, we inferred ploidy levels on 21 isolates from an outbreak of *Candida auris*.[30] In all contigs analysed, haploid was inferred as the most likely ploidy and multiploidy was rejected in all samples. These results are consistent with findings from the original study, although the identification of polyploidy in other isolates may be associated with multidrug-resistant phenotypes.[31]

Discussion

Analyses presented here provide further support for the use of genotype and allele frequency likelihoods in the analysis of NGS data.[5] Notably, we demonstrated how probabilistic estimates of population genetic parameters can be obtained in case of pooled sequencing data and short-read data from polyploid genomes. Additionally, we motivated the inference of SFS from allele frequency likelihoods as a direct way to infer demography from raw sequencing data.

`ngsJulia` offers new possibilities of software prototyping for custom analyses of NGS data for population genetic applications. Furthermore, it facilitates experimental design as it provides a platform to benchmark the efficacy of

Table 1. Confidence values to the correctly assigned ploidy vector and test for multiploidy. Likelihood-ratio test (LRT) statistics using different methods of incorporating allele frequencies are reported.

Allele frequency	LRT – ploidy	LRT - multiploidy
Folded	34.19	252.72
Ancestral	31.49	277.91
Sample	37.83	373.07

population genetic analysis from competing sequencing experiments. Finally, `ngsJulia` has accessible documentation and tutorials to inform users on the theory underpinning the implemented methods. We envisage that further improvements in `ngsJulia` will include the expansion of suitable input formats and data file types, and the compatibility with additional NGS data type, including from long-read sequencing experiments.[32]

Conclusions

In this study, we introduce `ngsJulia`, a series of templates and functions in `Julia` language to analyse NGS data for population genetic purposes. We present two implementations for the analysis of pooled sequencing data and polyploid genomes, with the inclusion of novel methods. `ngsJulia` is a suitable framework for prototyping new software and for custom population genetic analyses from NGS data.

Data availability

Underlying data

Simulated data and pipeline to reproduce all results presented here are available at https://doi.org/10.5281/zenodo.5886879.[20]

Software availability

- Source code available from: https://github.com/mfumagalli/ngsJulia
- Archived source code at time of publication: https://doi.org/10.5281/zenodo.5886879[20]
- License: Creative Commons Attribution 4.0 International Public License

Acknowledgements

We are thankful to Johanna Rhodes for assistance gathering sequencing data for *Candida auris*.

References

1. Marchi N, Schlichta F, Excoffier L: **Demographic inference.** *Curr. Biol.* 2021; **31**(6): R276–R279.
Publisher Full Text | Reference Source
2. Vitti JJ, Grossman SR, Sabeti PC: **Detecting natural selection in genomic data.** *Annu. Rev. Genet.* 2013; **47**(1): 97–120.
PubMed Abstract | Publisher Full Text
3. Uffelmann E, Huang QQ, Munung NS, *et al.*: **Genome-wide association studies.** *Nature Reviews Methods Primers.* Aug 2021; **1**(1): 59.
Publisher Full Text
4. Levy SE, Myers RM: **Advancements in next-generation sequencing.** *Annu. Rev. Genomics Hum. Genet.* 2016; **17**(1): 95–115.
PubMed Abstract | Publisher Full Text
5. Nielsen R, Paul JS, Albrechtsen A, *et al.*: **Genotype and snp calling from next-generation sequencing data.** *Nat. Rev. Genet.* June 2011; **12**(6): 443–451.
PubMed Abstract | Publisher Full Text | Reference Source
6. Li H: **A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data.** *Bioinformatics.* 09 2011; **27**(21): 2987–2993.
PubMed Abstract | Publisher Full Text
7. Van der Auwera GA, Carneiro MO, Hartl C, *et al.*: **From fastq data to high-confidence variant calls: The genome analysis toolkit best practices pipeline.** *Curr. Protoc. Bioinformatics.* 2013; **43**(1): 11.10.1–11.10.33.
PubMed Abstract | Publisher Full Text
8. Garrison E, Marth G: **Haplotype-based variant detection from short-read sequencing.** 2012.
Reference Source
9. Crawford J, Lazzaro B: **Assessing the accuracy and power of population genetic inference from low-pass next-generation sequencing data.** *Front. Genet.* 2012; **3**: 66. 1664-8021.
PubMed Abstract | Publisher Full Text
10. Fumagalli M: **Assessing the effect of sequencing depth and sample size in population genetics inferences.** *PLoS One.* 11 2013; **8**(11): 1–11.
PubMed Abstract | Publisher Full Text
11. Korneliussen TS, Albrechtsen A, Nielsen R: **Angsd: analysis of next generation sequencing data.** *BMC Bioinformatics.* 2014; **15**(1): 356.
PubMed Abstract | Publisher Full Text
12. Fumagalli M, Vieira FG, Linderoth T, *et al.*: **ngsTools: methods for population genetics analyses from next-generation sequencing data.** *Bioinformatics.* May 2014; **30**(10): 1486–1487.
PubMed Abstract | Publisher Full Text | Reference Source
13. Lou RN, Jacobs A, Wilder A, Therkildsen NO: **A beginner's guide to low-coverage whole genome sequencing for population genomics.** *Mol. Ecol.* 2021; **30**: 5966–5993
Publisher Full Text
14. Schlötterer C, Tobler R, Kofler R, Nolte V: **Sequencing pools of individuals — mining genome-wide polymorphism data without big funding.** *Nat. Rev. Genet.* Nov 2014; **15**(11): 749–763.
PubMed Abstract | Publisher Full Text
15. Kofler R, Pandey RV, Schlötterer C: **PoPoolation2: identifying differentiation between populations using sequencing of pooled DNA samples (Pool-Seq).** *Bioinformatics.* 10 2011; **27**(24): 3435–3436.
PubMed Abstract | Publisher Full Text
16. Raineri E, Ferretti L, Esteve-Codina A, *et al.*: **Snp calling by sequencing pooled samples.** *BMC Bioinformatics.* Sep 2012; **13**(1): 239.
PubMed Abstract | Publisher Full Text
17. Bezanson J, Edelman A, Karpinski S, *et al.*: **Julia: A fresh approach to numerical computing.** *SIAM Rev.* 2017; **59**(1): 65–98.
Publisher Full Text
18. Sato K, Tsuyuzaki K, Shimizu K, *et al.*: **Cellfishing.jl: an ultrafast and scalable cell search method for single-cell rna sequencing.** *Genome Biol.* Feb 2019; **20**(1): 31.
PubMed Abstract | Publisher Full Text

19. Li H, Handsaker B, Wysoker A, *et al.*: **The Sequence Alignment/Map format and SAMtools.** *Bioinformatics.* 06 2009; **25**(16): 2078–2079.
PubMed Abstract | Publisher Full Text

20. Fumagalli M: **ngsjulia: population genetic analysis of next-generation dna sequencing data with julia language.** *Zenodo.* 2022.
Publisher Full Text

21. McKenna A, Hanna M, Banks E, *et al.*: **The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data.** *Genome Res.* sep 2010; **20**(9): 1297–303.
Publisher Full Text | Reference Source

22. Soraggi S, Rhodes J, Altinkaya I, *et al.*: **Hmmploidy: inference of ploidy levels from short-read sequencing data.** *Peer Commun J.* 2022; **2**: e60.
Publisher Full Text | Reference Source

23. Kim SY, Lohmueller KE, Albrechtsen A, *et al.*: **Estimation of allele frequency and association mapping using next-generation sequencing data.** *BMC Bioinformatics.* Jun 2011; **12**(1): 231.
PubMed Abstract | Publisher Full Text

24. Nielsen R, Korneliussen T, Albrechtsen A, *et al.*: **Snp calling, genotype calling, and sample allele frequency estimation from new-generation sequencing data.** *PLoS One.* 07 2012; **7**(7): 1–10.
PubMed Abstract | Publisher Full Text

25. Ewens WJ: **The sampling theory of selectively neutral alleles.** *Theor. Popul. Biol.* 1972; **3**(1): 87–112.
PubMed Abstract | Publisher Full Text | Reference Source

26. Bielski CM, Zehir A, Penson AV, *et al.*: **Genome doubling shapes the evolution and prognosis of advanced cancers.** *Nature Genetics.* Aug 2018; **50**(8): 1189–1195.
Publisher Full Text

27. Fumagalli M, Vieira FG, Korneliussen TS, *et al.*: **Quantifying population genetic differentiation from next-generation sequencing data.** *Genetics.* 2013; **195**(3): 979–992.
PubMed Abstract | Publisher Full Text | Reference Source

28. Fracassetti M, Griffin P. C, Willi Y., *et al.*: **Validation of pooled whole-genome re-sequencing in arabidopsis lyrata.** *PLoS One.* 2015; **10**(10): 1–15.
Publisher Full Text | Reference Source

29. Koboldt D. C, Zhang Q, Larson D. E, *et al.*: **Varscan 2: Somatic mutation and copy number alteration discovery in cancer by exome sequencing.** *Genome Research.* 2012; **22**(3): 568–576.
Publisher Full Text | Reference Source

30. Rhodes J., Abdolrasouli A., Farrer R. A, *et al.* **Genomic epidemiology of the uk outbreak of the emerging human fungal pathogen candida auris**. *Emerging Microbes & Infections*. 2018; **7**(1): 1–12.
PubMed Abstract | Publisher Full Text | Reference Source

31. Navalkele B. D, Revankar S, Chandrasekar P.: **Candida auris: a worrisome, globally emerging pathogen.** *Expert Review of Anti-infective Therapy*. 2017; **15**(9): 819–827.
PubMed Abstract | Publisher Full Text | Reference Source

32. Logsdon GA, Vollger MR, Eichler EE: **Long-read human genome sequencing and its applications.** *Nat. Rev. Genet.* Oct 2020; **21**(10): 597–614.
PubMed Abstract | Publisher Full Text

Open Peer Review

Current Peer Review Status: ? ✔ ? ?

Version 3

Reviewer Report 11 October 2023

https://doi.org/10.5256/f1000research.150492.r202079

?

Jens Léon
University of Bonn, Bonn, Germany
Michael Schneider
Forschungsinstitut fur Biologischen Landbau, Frick, Aargau, Switzerland

First of all, I have to write that we are working on a method to estimate the allele frequency in crops by pool sequencing. Therefore, we are very interested in this new methodology. In this review our research group have a look on the manuscript. We are researchers dealing with population and quantitative genetics of plants and other are dealing with bioinformatics.

We recognized that the examples and sources for the testing are rather complex. The authors explain their probes to be haploid, usually diploid but also polyploid. They state that the samples of the organisms are mostly biallelic but sometimes also tri-allelic. We expect that in a population we usually have multiple alleles. The authors are also dealing with pooled samples. But for researches dealing with theses probes, it is rather confusing to understand the restrictions. Regarding a divers readership, I would like to recommend that the authors clearly define their samples first (haploid, diploid or polyploid and whether biallelic or multi-allelic loci are expected). When dealing with pooled samples the readership should know about the expectation of the samples (e.g. pool of sperms or pollen grains are different from a pool of cells from an individuum). Analysing populations with pooled NGS, it would be important to know how the software deals with a highly heterogenous material, as several individuals in a population. Although the authors tried to group their methods, it is still unclear which expectations the authors have to the samples.

It would be helpful to provide a code example of creating the mpileup file.

Why do you need to provide a new tool to call SNPs? The three tools in the introduction are all solid options
and can give allele frequency calls, too. Adding the info on the expected allele frequency is helpful - but the effect of this additional information
would be worth demonstrating. E.g., by showing the accuracy of recalling the correct allele

frequency with and without this term. Besides,
how would someone with a heterogeneous population set this value?

It would be helpful to illustrate the actual distribution of reads across the genome. It is hard to imagine that with a uniform coverage of 0.5,
which is not uncommon these days to receive from a WGS run, a good recall rate of the allele frequency can be achieved. It would, therefore, be helpful to understand the assumed input data better and provide the lambda value of the Poisson model for read simulations.

There is no information provided about the recall rate of simulated SNPs. How many SNPs were detected on the different sequencing depth levels,
and how many were falsely discovered? This is a major problem in the other named variant callers at very low sequencing depth levels, as discussed
in this manuscript, and it would be of high value to compare the accuracy and sensitivity of the new ngsPool approach and the "old" once, like bcftools
and GATK.

Real Arabidopsis data comparing ngsPool to VarScan. As previously mentioned, why not compare also to GATK, FreeBayes, and bcftools? VarScan to my knowledge is not one of the top tools in terms of variant detection.

To give this manuscript more relevance and make the decision to use it or not more accessible for the reader, I would encourage the authors to provide the information if this new package performs more precisely than common tools, if not, or under which circumstances it does. If the accuracy is not improved, what are the other benefits? Time-saving, user-friendly output.

Figure caption of Figure 1: the order in the graphics is not the same as the one in the description. That can be changed easily and helps the reader.

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Population and quantitative genetics of crops, Plant breeding, cereals

We confirm that we have read this submission and believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however we have significant reservations, as outlined above.

Reviewer Report 21 September 2023

https://doi.org/10.5256/f1000research.150492.r202073

?

Nikolay Oskolkov
Department of Biology, National Bioinformatics Infrastructure Sweden, Science for Life Laboratory, Lund University, Lund, Sweden

The authors suggest a population genomics tool written in Julia programming language. They demonstrate the performance of the tool on simulated and real pooled sequencing data. The manuscript is well written and technical details of the tool are well explained. In addition, documentation is provided together with the tool.

I believe, it is potentially interesting to approach some bioinformatics analyses (such as population genomics) with Julia. However, I am still missing a clear motivation for re-writing e.g. genotype likelihoods and SFS computation in Julia. There are many popgen tools that are fast and accurate enough, e.g. ANGSD, so there should be a clear advantage of ngsJulia in some way compared to other tools. If the motivation is the outstanding speed or accuracy of ngsJulia, it should be clearly demonstrated. So my main concern is the lack of comparison of ngsJulia with other popgen tools.

Next, I would not advertise ngsJulia as a "*population genetic analysis of next-generation sequencing data*", as from the text I got an impression that this was a quite pool-seq-specific tool. If ngsJulia is more general than a pool-seq data tool, it should be demonstrated and compared with other tools. Otherwise, I would suggest to change the title of the manuscript and make it less general and more specific. For example, accounting for polyploidy is a strength of ngsJulia, perhaps this should be emphasized more in the manuscript.

I found it interesting and novel to determine the optimal K in the 1/f^K approximation of SFS profile with Kullback-Leibler optimization. However, as it seems to me from the Figure 5, this implementation in ngsJulia only works for D ~ 5 and large sample sizes. Perhaps other tools will not do a great job either, but as I mentioned above, I am missing the comparison with other tools.

Minor comment:
Page 7 I would encourage the authors to make the command line for ngsPool.jl more self-

explanatory. More specifically, the "*–lrtSnp 6.64*" flag is hard to grasp. Perhaps re-writing it in a way so that it becomes clear that 6.64 corresponds to p-value = 0.01, and replacing the flag "–lrtSnp 6.64" with "--pvalue 0.01" would be possible?

Is the rationale for developing the new software tool clearly explained?
Partly

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: bioinformatics, genomics, medical genetics, ancient DNA

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reviewer Report 14 August 2023

https://doi.org/10.5256/f1000research.150492.r187509

Stéphane De Mita
Institut national de la recherche agronomique (INRAE), Paris, France

After considering the updated version and using the new documentation, I managed to run some tests successfully. However, I couldn't understand the return value of calcGenoLike without extrapolating from what it done in the triploid case in the tutorial.

Is the rationale for developing the new software tool clearly explained?

Partly

Is the description of the software tool technically sound?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Population genomics, phytopathology, programming (Python, C, C++).

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 2

Reviewer Report 21 February 2023

https://doi.org/10.5256/f1000research.141471.r160519

?

Stéphane De Mita
[1] Institut national de la recherche agronomique (INRAE), Paris, France
[2] Institut national de la recherche agronomique (INRAE), Paris, France

The article "ngsJulia: population genetic analysis of next-generation DNA sequencing data with Julia language" by Alex Mas-Sandoval and colleagues provides a library for the Julia addressing management of NGS data (provided in the PileUp format), SNP and genotype calling allowing for arbritary and dataset-varying ploidy, and maximum-likelihood parameter estimation.

I've really discovered the Julia language while reviewing this article and I agree that it makes sense to use this language for developing tools for the analysis of NGS data. Providing a toolkit to

perfom SNP and genotype calling under a probabilistic models makes much sense and can be useful for the community. The ngsPool application provided can be of widespread use while ngsPloidy can be a very precious tool for specific systems which lack proper software right now.

Here is a list of random typos or detail-level suggestions:

- "polidy" in the "Revision" box.
- Methods: you should add Combinatorics as a requirement.
- I am not sure to understand the header "Operation".
- One or two sentences, presenting biological systems where the study of multiploidy (or in general MLE of ploidy) is relevant, would help.
- I am not sure giving an example of a R command line for simulating data is a good choice. For a Julia package I'd like to see some Julia code example, although it is not absolutely necessary.
- Discussion: rewrite: "is highly applicable in educational contexts"

I do not get this sentence at all: "*[ngsJulia] allows for efficient testing of experimental designs and, therefore, would be beneficial for initial planning of any sequencing experiments.*"

My big issue with the paper, which is confirmed by a first look at the package, is that the content of the ngsJulia library isn't described, even succinctly. There should be an API and the article should point to it.

In my view there is still need for more documentation. Tutorials are great and (speaking by experience) they are often essential, but at some point you need for some more rigourous reference documentation (API for the library, CLI for the applications). The Application Programmer Interface should be documented so that the reading planning to write a script using the library can know what functions are available, what input they expect, and if possible what errors can be expected. This remark also applies to the Command Line Interface of the two applications ngsPloidy and ngsPool. The help page produces an option summary, which is fine, but the meaning of the options isn't documented.

Below I report the results of my testing of the three components of the software.

*** ngsPloidy ***

The tutorial is a good resource to start, but some points need clarification, and a more rigourous reference manual of the command-line interface would help a lot.

I am not sure that it is a good idea to start by defining environment variables, which make the commands much less friendly. At least, do mention that you are defining environment variables so that unaware users can know where to look at if they need more information. Anyway the julia command should be available is it is installed in the environment which should be considered to be the default. By the way, "Julia language" is a poor choice of words.

I am sorry, but I didn't get at all what "genotype probabilities" where and whether and why would I need to to create them. I have no idea what writePars is doing and -p and -s arguments aren't clearly documented.

There is a "variabile" (instead of "variable") in the in simulMpileup.R manual. The simulation script needs its own documentation.

Personally, I would think that either bam files containing aligned reads, or VCF, would be a better choice than PileUp, because users are more likely to already have those.

Suggestions: you could print progress information while running, and displays error messages in stdout instead of stderr (especially useful since you advice users to redirect results which are printed in stdout.

I tested ngsPloidy using data from my own read simulator. I thought it was a good idea to bring data from an independent source, and also I am more comfortable using it. It generated data from diploid and tetraploid organisms, using a coalescent model to draw allelic frequencies.

Then I ran this type of commands:
$ julia ../ngsJulia/ngsPloidy/ngsPloidy.jl --fin diplo.plp.gz --nSamples 20 > diplo.out

Running times were (tens of minutes up to an hour for rather small simulated datasets). I suspect that I should have dropped non-varying sites from the PileUp (since I didn't incorporate errors, 90% of my sites are completely fixed).

When I simulated only diploid or tetraploid individuals, they are completely inferred (although in some parameter settings I had tetraploid individuals estimated as a mixture of tetra- and pentaploid individuals). When I merge diploid and tetraploid individuals in a single PileUp, the software tended to estimate a mixture of diploid and triploid individuals. So I suspect that I am doing something wrong. Most likely I miss an important step of the procedure and the documentation might be to blame.

*** ngsPool ***

Among the command line arguments, by "LRT", do you mean "threshold"?

I got an exception when the pileup file didn't contain reference bases. I understand that this should be an error but it could be displayed in a more accessible manner.

Some test results are reported as "Inf" or "-Inf" which is also not necessarily clear.

I was not totally clear when --lrt* arguments should be used.

I got a problem to understand the difference between maf, saf_MLE and saf_E? It is also a problem with the article, where the logic is mentioned by not really explained.

I ran a simulation with a 20,000 bp genome, 20 diploid individuals, 10% of sites with diallelic polymorphism, 10,000 reads of 50 bp (so an average depth of 25X).

I ran:
```
# julia ../ngsJulia/ngsPool/ngsPool.jl --fin data.plp.gz \
        --fout results.txt.gz  --lrtSnp 6.64
```

It took ~ 4:30 minutes and the results are fitting the simulation parameters. All sites called as SNPs are correct and only occasionnaly a few number (~1/1000) of "real" SNPs aren't called, and whose are always singletons. There is also a good agreement between the 'real' and estimated maf.

Using the --nChroms option gives similar results and all estimated frequencies are correct.

*** ngsJulia ***

An API reference was badly needed. I tried to start writing a Julia script analysing one of the datasets used for previous testing and didn't get very far. My lack of knowledge of Julia didn't help, for sure.

The last paragraph of the readme "In general, ngsJulia provides templates..." should come first.

Is it normal that we need to import Combinatorics even if we are not directly using it (i.e. I reckon that ngsJulia should import it itself).

Almost instantly, I have the following line crashing:
```
nucleoLikes = [calcGenoLike(myReads, [i], 1) for i=1:4]
```

The error message is: ERROR: LoadError: BoundsError: attempt to access SubString{String}
at index [2]

I thought it was because I was trying to pass a site (represented by a Reads object, I assume), containing a single read base. But in the end it is not this (sites with more bases are not accepted either). Obviously I am making a very basic error but the lack of documentation doesn't help me.

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Population genomics, phytopathology, programming (Python, C, C++).

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 06 Jul 2023

Matteo Fumagalli

Here is a list of random typos or detail-level suggestions:

- "polidy" in the "Revision" box.

Unfortunately we cannot modify previous responses to reviewers, but this typo is not present in the manuscript itself.

- Methods: you should add Combinatorics as a requirement.

Added.

- I am not sure to understand the header "Operation".

This is a required subheader for Software Application submissions in F1000Research (https://f1000research.com/for-authors/article-guidelines/software-tool-articles).

- One or two sentences, presenting biological systems where the study of multiploidy (or in general MLE of ploidy) is relevant, would help.

We now cite Bielski et al. 2018 as an example of multiploidy in tumor genomes. Soraggi et al. 2022 is cited as a framework for MLE of ploidy.

- I am not sure giving an example of a R command line for simulating data is a good choice. For a Julia package I'd like to see some Julia code example, although it is not absolutely necessary.

We agree that ideally all code should be in Julia, especially for simulating data. We plan to provide a Julia script for this purpose in the near future.

- Discussion: rewrite: "is highly applicable in educational contexts"

We rephrased the sentence as "Finally, ngsJulia has accessible documentation and tutorials to inform users on the theory underpinning the implemented methods."

I do not get this sentence at all: "*[ngsJulia] allows for efficient testing of experimental designs and, therefore, would be beneficial for initial planning of any sequencing experiments.*"

We rephrased the sentence as "[ngsJulia] facilitates experimental design as it provides a platform to benchmark the efficacy of population genetic analysis from competing sequencing experiments."

My big issue with the paper, which is confirmed by a first look at the package, is that the content of the ngsJulia library isn't described, even succinctly. There should be an API and the article should point to it.
In my view there is still a need for more documentation. Tutorials are great and (speaking by experience) they are often essential, but at some point you need for some more rigourous reference documentation (API for the library, CLI for the applications). The Application Programmer Interface should be documented so that the reading planning to write a script using the library can know what functions are available, what input they

expect, and if possible what errors can be expected. This remark also applies to the Command Line Interface of the two applications ngsPloidy and ngsPool. The help page produces an option summary, which is fine, but the meaning of the options isn't documented.

We now provide extensive documentation of APIs and CLIs at https://ngsjulia.readthedocs.io/en/latest/
We also provide docstring documentation for each function.

Below I report the results of my testing of the three components of the software.
*** ngsPloidy ***
The tutorial is a good resource to start, but some points need clarification, and a more rigourous reference manual of the command-line interface would help a lot.

We agree and a new documentation has been provided in the readthedocs webpage.

I am not sure that it is a good idea to start by defining environment variables, which make the commands much less friendly. At least, do mention that you are defining environment variables so that unaware users can know where to look at if they need more information. Anyway the julia command should be available is it is installed in the environment which should be considered to be the default. By the way, "Julia language" is a poor choice of words.

We now assume that Julia is available and specify that we define an environment variable for ngsJulia. References to "Julia language" have been removed.

I am sorry, but I didn't get at all what "genotype probabilities" where and whether and why would I need to to create them. I have no idea what writePars is doing and -p and -s arguments aren't clearly documented.

At the beginning of the documentation for ngsPloidy, we now briefly mention the model used to infer ploidy levels and refer to the paper for more details. "In a nutshell, the algorithm calculates the likelihood of ploidy levels from genotype likelihoods and genotype priors. The latter can be estimated either from the data or, with limited sample size, from a population genetic model." Additionally, in the documentation before the first example using writePars.R, we state "With limited sample size we can use a different estimation of population allele frequency which will be constant across all sites. In case of limited sample size, firstly we need to create a file containing genotype probabilities. We also need to provide the probability of the major allele being ancestral, as this information will be used in case of limited sample size. These probability files can be generated using the following R script [...]". We also provide more details in a separate page https://ngsjulia.readthedocs.io/en/latest/aux/

There is a "variabile" (instead of "variable") in the in simulMpileup.R manual. The simulation script needs its own documentation

We fixed the typo and we now provide documentation for the script on the

readthedocs webpage at https://ngsjulia.readthedocs.io/en/latest/aux/**.**

Personally, I would think that either bam files containing aligned reads, or VCF, would be a better choice than PileUp, because users are more likely to already have those.

We agree and we plan to implement a parser of BAM/CRAM files in Julia in the future. We cannot use VCF as they do not contain reads information to calculate genotype likelihoods from scratch. As mpileup files are easily generated using samtools (see https://www.htslib.org/doc/samtools-mpileup.html), we believe that users should be able to produce these files.

Suggestions: you could print progress information while running, and displays error messages in stdout instead of stderr (especially useful since you advice users to redirect results which are printed in stdout.

Both CLIs have –verbose and –printSites options (now documented) to regulate the amount and frequency of progress information while running. Both CLIs print error messages in stdout.

I tested ngsPloidy using data from my own read simulator. I thought it was a good idea to bring data from an independent source, and also I am more comfortable using it. It generated data from diploid and tetraploid organisms, using a coalescent model to draw allelic frequencies.
Then I ran this type of commands:
$ julia ../ngsJulia/ngsPloidy/ngsPloidy.jl --fin diplo.plp.gz --nSamples 20 > diplo.out
Running times were (tens of minutes up to an hour for rather small simulated datasets). I suspect that I should have dropped non-varying sites from the PileUp (since I didn't incorporate errors, 90% of my sites are completely fixed).

Doing SNP calling would drastically reduce the running time and most of the analyses presented can be performed on called SNPs only.

When I simulated only diploid or tetraploid individuals, they are completely inferred (although in some parameter settings I had tetraploid individuals estimated as a mixture of tetra- and pentaploid individuals). When I merge diploid and tetraploid individuals in a single PileUp, the software tended to estimate a mixture of diploid and triploid individuals. So I suspect that I am doing something wrong. Most likely I miss an important step of the procedure and the documentation might be to blame.

We provide performance results on estimating ploidy levels at different scenarios in the paper. Inferring complex ploidy patterns at certain sequencing experimental settings is a challenging task. Results in the paper should be able to show at which conditions ploidy levels tend to be correctly inferred.

*** ngsPool ***
Among the command line arguments, by "LRT", do you mean "threshold"?

LRT stands for "likelihood ratio test". We now describe the options as "chisquare value for SNP calling" to be consistent with the description in ngsPloidy.

I got an exception when the pileup file didn't contain reference bases. I understand that this should be an error but it could be displayed in a more accessible manner.

We are now explicit in the documentation that pileup must include information in the reference bases.

Some test results are reported as "Inf" or "-Inf" which is also not necessarily clear.

These values indicate very high or low values for the specific metric or statistic that has been calculated. We believe that such representation is in line with other programming languages (R, python) and users should be able to understand its significance.

I was not totally clear when --lrt* arguments should be used.

–lrt arguments are used for SNP calling, for including only biallelic SNPs, or for excluding triallelic SNPs. This argument calculates a statistical test (likelihood ratio test, lrt) to perform such calling as explained in the paper.

I got a problem to understand the difference between maf, saf_MLE and saf_E? It is also a problem with the article, where the logic is mentioned by not really explained.

Three estimators of allele frequencies are implemented: a population maximum likelihood estimate (MLE) from unknown sample size ("maf" in the results) and the expected value (saf_E) and MLE (saf_MLE) from known sample size. Details on their equations are provided in the paper:

Briefly, when the sample size is not known (which can happen in pooled-sequencing data), a simple MLE estimator of the minor allele frequency (maf) is implemented assuming that haploid state. When the sample size is known, we calculate sample allele frequency likelihoods (saf). From these likelihoods, we can either calculate the expected value (saf_E) or the maximum likelihood estimate (saf_MLE).

I ran a simulation with a 20,000 bp genome, 20 diploid individuals, 10% of sites with diallelic polymorphism, 10,000 reads of 50 bp (so an average depth of 25X).
I ran:

```
# julia ../ngsJulia/ngsPool/ngsPool.jl --fin data.plp.gz \
        --fout results.txt.gz  --lrtSnp 6.64
```

It took ~ 4:30 minutes and the results are fitting the simulation parameters. All sites called as SNPs are correct and only occasionnaly a few number (~1/1000) of "real" SNPs aren't called, and whose are always singletons. There is also a good agreement between the 'real' and estimated maf.
Using the --nChroms option gives similar results and all estimated frequencies are correct.

Thank you for checking its performance on a different simulated data set.

*** ngsJulia ***
An API reference was badly needed. I tried to start writing a Julia script analysing one of the datasets used for previous testing and didn't get very far. My lack of knowledge of Julia didn't help, for sure.

We now provide documentation for all APIs implemented.

The last paragraph of the readme "In general, ngsJulia provides templates..." should come first.

The readme has been changed.

Is it normal that we need to import Combinatorics even if we are not directly using it (i.e. I reckon that ngsJulia should import it itself).

ngsJulia now imports all required packages. Additionally, the user simply needs to run "include('ngsJulia.jl');" now.

Almost instantly, I have the following line crashing:
  nucleoLikes = [calcGenoLike(myReads, [i], 1) for i=1:4]
The error message is: ERROR: LoadError: BoundsError: attempt to access SubString{String}
  at index [2]
I thought it was because I was trying to pass a site (represented by a Reads object, I assume), containing a single read base. But in the end it is not this (sites with more bases are not accepted either). Obviously I am making a very basic error but the lack of documentation doesn't help me.

We now provide extensive documentation and examples on the use of all APIs including "calcGenoLike" to facilitate their usage. In this specific example (which works on the example provided in the documentation), the docstring documentation for this function is:

We finally fixed minor typos, such as "et al." in the text.

Competing Interests: No competing interests were disclosed.

Version 1

Reviewer Report 13 April 2022

https://doi.org/10.5256/f1000research.114499.r128232

?

Lindsay Clark
[1] Roy J. Carver Biotechnology Center, University of Illinois at Urbana-Champaign, Urbana, IL, USA
[2] Roy J. Carver Biotechnology Center, University of Illinois at Urbana-Champaign, Urbana, IL, USA
[3] Roy J. Carver Biotechnology Center, University of Illinois at Urbana-Champaign, Urbana, IL, USA

The manuscript by Mas-Sandoval et al. describes ngsJulia, a collection of functions and scripts in Julia and R for estimating genotype likelihoods from short read DNA sequencing data, as well as estimating allele frequencies in pooled samples, and estimating ploidy where it is unknown. It seems that the goal is also to enable biologists proficient in Julia to develop their own custom genotype calling and population genetic analysis scripts starting with the provided functions for estimating genotype likelihoods. As it is, the manuscript and software suffer from two major flaws: (1) the software and its outputs are not well documented for users or developers, and (2) testing is only performed on simulated data, not empirical data.

The software documentation is not very approachable as it currently is. End-to-end tutorials with a real dataset (or something closely resembling a real dataset) are needed for all applications. How are the output files formatted and how might the user further process those files to address biological questions? For example, it seems that one could use ngsJulia to generate a matrix of genotype calls across samples and loci, starting from a set of mpileup files, but from the documentation I have no idea how to accomplish this task.

There are eleven different functions for estimating genotype log likelihoods, depending on ploidy, with four different functions for haploids. This strikes me as poor software design. At minimum the eight calcGenoLogLikeX_MajorMinor functions could be collapsed into one, with ploidy as an added argument to the function. Having a single function would make it much easier to update the code and ensure that there aren't any bugs that prevent it from working consistently across ploidy levels. The hard-coding of tuples for every possible genotype, and the repetitive if-else statements, could be replaced with a more programmatic approach. The programmatic approach with ploidy as a function argument would also enable the software to be used for organisms with ploidy greater than octoploid.

If I understand correctly, the SNP calling algorithm only seems to work on haploid samples, which would exclude it from being used in most studies. Please clarify.

The estimation of ploidy from sequencing data using the ngsPloidy is a valuable addition. As an author of other software for marker analysis in polyploids, I frequently hear requests for such functionality, so I expect ngsPloidy to get a lot of use. It would be helpful to include a comparison to other tools such as ploidyNGS, ConPADE, and/or SuperMASSA. Additionally, the algorithm is only tested on a simulated dataset, and I would like to see it tested on an empirical dataset (with flow cytometry used as ground truth), both to see how well it scales up to thousands of markers, and how well it deals with potentially messy data.

The authors use the term "aneuploidy" incorrectly. Aneuploidy refers to different chromosomes

having different copy numbers within one individual. The authors seem to mean different individuals within a population having different ploidies. “Multiploid” is one term for this although depending on the field it is sometimes used as a synonym for “polyploid”.

How should the user estimate the shape of the site frequency spectrum and the effective population size, which are required inputs for ngsPloidy?

I tried “Case A” in the ngsPloidy tutorial. It took me a few minutes to figure out the (undocumented) contents of test.A.txt. I assume that columns 6-25 are the true genotypes, column 5 is the allele frequency, and column 26 is the sample allele frequency. The tutorial otherwise worked. However, I didn’t get anything useful when I added the --callGeno flag.

Similarly to ngsPloidy, ngsPool is only tested on simulated data.

Minor comments:

- There seems to be a missing right bracket on the left side of equation 2.
- The Figure 2 caption needs to more explicitly define the abbreviations MLE and SAF.
- All of the code to reproduce the analysis in the paper is provided, although there is no documentation explaining the code or indicating which scripts go with which figures.

Is the rationale for developing the new software tool clearly explained?
Partly

Is the description of the software tool technically sound?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Bioinformatics, polyploidy, population genetics

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have

significant reservations, as outlined above.

Author Response 22 Nov 2022

Matteo Fumagalli

The manuscript by Mas-Sandoval et al. describes ngsJulia, a collection of functions and scripts in Julia and R for estimating genotype likelihoods from short read DNA sequencing data, as well as estimating allele frequencies in pooled samples, and estimating ploidy where it is unknown. It seems that the goal is also to enable biologists proficient in Julia to develop their own custom genotype calling and population genetic analysis scripts starting with the provided functions for estimating genotype likelihoods. As it is, the manuscript and software suffer from two major flaws: (1) the software and its outputs are not well documented for users or developers, and (2) testing is only performed on simulated data, not empirical data.

Thank you for reviewing our manuscript. We addressed all concerns by improving the applicability and documentation of our software and by applying it to empirical data.

The software documentation is not very approachable as it currently is. End-to-end tutorials with a real dataset (or something closely resembling a real dataset) are needed for all applications. How are the output files formatted and how might the user further process those files to address biological questions? For example, it seems that one could use ngsJulia to generate a matrix of genotype calls across samples and loci, starting from a set of mpileup files, but from the documentation I have no idea how to accomplish this task.

We improved the documentation as requested in the specific comments (see below). Examples from simulated mpileup files under various conditions are provided and now clarified. In our response, we clarified that ngsPloidy does not provide a matrix of genotype calls but rather estimates ploidy levels.

There are eleven different functions for estimating genotype log likelihoods, depending on ploidy, with four different functions for haploids. This strikes me as poor software design. At minimum the eight calcGenoLogLikeX_MajorMinor functions could be collapsed into one, with ploidy as an added argument to the function. Having a single function would make it much easier to update the code and ensure that there aren't any bugs that prevent it from working consistently across ploidy levels. The hard-coding of tuples for every possible genotype, and the repetitive if-else statements, could be replaced with a more programmatic approach. The programmatic approach with ploidy as a function argument would also enable the software to be used for organisms with ploidy greater than octoploid.

We agree with this comment and we now replace all those functions into a unique one that takes ploidy as input parameter (see calcGenoLike function in file functions.jl). We also replaced all if-else statements with a more programmatic approach which makes ngsPloidy (see file ngsPloidy.jl).

If I understand correctly, the SNP calling algorithm only seems to work on haploid samples, which would exclude it from being used in most studies. Please clarify.

The SNP calling algorithm relies on either a likelihood ratio test with the null hypothesis being the pooled allele frequency equal to 0 (similar to Kim et al. 2011 Bioinformatics), or from setting a threshold on the estimated pooled allele frequency. Therefore, it is applicable to all ploidy levels.

The estimation of ploidy from sequencing data using the ngsPloidy is a valuable addition. As an author of other software for marker analysis in polyploids, I frequently hear requests for such functionality, so I expect ngsPloidy to get a lot of use. It would be helpful to include a comparison to other tools such as ploidyNGS, ConPADE, and/or SuperMASSA. Additionally, the algorithm is only tested on a simulated dataset, and I would like to see it tested on an empirical dataset (with flow cytometry used as ground truth), both to see how well it scales up to thousands of markers, and how well it deals with potentially messy data.

This manuscript (submitted as Software Tool Article) aims at proposing a new implementation rather than a new methodology. Additionally, the likelihood function to estimate ploidy levels partly mirrors the approach taken by a parallel study (https://www.biorxiv.org/content/10.1101/2021.06.29.450340) where we provide extensive comparisons with competing algorithms (including ngsPloidy). Therefore, a comprehensive benchmarking of ngsPloidy against alternative software would be outside the scope of this paper.
As suggested, we applied ngsPloidy to a real empirical data of isolates from an outbreak of *Candida auris* (Rhodes et al. 2018). Whilst polyploidy could be associated with multidrug-resistant phenotypes, we inferred haploidy for all samples and contigs, in line with findings from the original study. We added a paragraph on both methods and results sections to describe this new analysis.

The authors use the term "aneuploidy" incorrectly. Aneuploidy refers to different chromosomes having different copy numbers within one individual. The authors seem to mean different individuals within a population having different ploidies. "Multiploid" is one term for this although depending on the field it is sometimes used as a synonym for "polyploid".

We replaced all occurrences of "aneuploidy" in the text and documentation with "multiploidy".

How should the user estimate the shape of the site frequency spectrum and the effective population size, which are required inputs for ngsPloidy?

The need to specify the shape of the site frequency spectrum and the effective population size is only required when the sample size is too low to have a meaningful estimate of the population allele frequency. Whilst these parameters are not known a prior for nonmodel species, in practice we found that, as long as values are within a reasonable range and the coverage is sufficient, there is not a significant bias associated with inaccurate settings of

these parameters. In fact, they simply determine the prior probability of sampling allele frequencies and, for most species and populations, this distribution tends to have more mass at lower frequency values.

I tried "Case A" in the ngsPloidy tutorial. It took me a few minutes to figure out the (undocumented) contents of test.A.txt. I assume that columns 6-25 are the true genotypes, column 5 is the allele frequency, and column 26 is the sample allele frequency. The tutorial otherwise worked. However, I didn't get anything useful when I added the --callGeno flag.

We now indicate the contents of all columns of the simulated file. The -callGeno flag does not output called genotypes but rather estimate ploidy by first assigning (calling) genotypes, and thus disabling the integrating over unknown genotypes. This is now clarified in the documentation.

Similarly to ngsPloidy, ngsPool is only tested on simulated data.

We now also present an application of ngsPool to empirical data. We reanalysed genomic data from *Arabidopsis lyrata* (Fracassetti et al. 2015). In this study, authors generated data both by genotype-by-sequencing and by pooled-sequencing, with the former providing ground-truth values for genotype and allele calls. We found that estimates of minor allele frequencies using ngsPool yield a lower RMSE (root mean squared error) than estimates from Varscan across all SNPs analysed. We added a paragraph both in the methods and results sections to describe these new analyses.

Minor comments:
There seems to be a missing right bracket on the left side of equation 2.

Fixed.

The Figure 2 caption needs to more explicitly define the abbreviations MLE and SAF.

We added the acronyms SAF and MLE in the caption.

All of the code to reproduce the analysis in the paper is provided, although there is no documentation explaining the code or indicating which scripts go with which figures.

In https://github.com/mfumagalli/ngsJulia/tree/master/paper we provide all code to replicate the analyses in the paper. As now indicated, ploidy/do.sh would replicate the simulations and estimation of ploidy variation. We now clarify that in the pool and pool/plots folder each folder/script corresponds to the specific figure in the paper.

Competing Interests: No competing interests were disclosed.