"PhaseImpute" an NF-Core pipeline for genetic imputation

Louis LE NEZET[1], NF-Core community, Pascale QUIGNON[1] and Catherine ANDRE[1]

1 UR1 - CNRS, ERL Inserm U1305 - UMR6290 IGDR (Institute of Genetics and Development of Rennes) - 35000 Rennes, France

Corresponding Author: louis.lenezet@univ-rennes.fr

Keywords

New Generation Sequencing (NGS); Genetic imputation; Nextflow; Open-Science

Abstract

Genotype and low coverage sequencing data provide cost-effective avenues for genomic research but inherently exhibit the limitations of being low resolution and low quality respectively. This sparsity in genetic data poses challenges, particularly in non-model organisms lacking species-specific phased panels, necessitating accurate haplotype phasing for effective downstream analyses. Genetic imputation serves as a valuable tool to supplement these methods by completing missing data and assigning probabilities to variant observations, thus enhancing data resolution and quality. However, ensuring the reliability of imputed genotypes requires rigorous tool validation and thorough exploration of parameter impacts (e.g. sliding window size, effective diploid population size, number of burn-in iterations, …), critical for maximising the utility of these techniques in genomic studies.

To tackle these challenges, we introduce a NF-Core compliant pipeline tailored for phasing, imputation, and validation of the imputed data. Our pipeline equips users with advanced genomic analysis tools for phasing and imputation analysis. This enables them to leverage the full potential of their genetic data by filling in missing information, harmonising datasets, and enhancing the resolution of genetic analyses. By adhering to NF-Core guidelines, our pipeline ensures users have access to a suite of high standard, versioned, and rigorously tested up-to-date tools, developed within a large and helpful community - a crucial aspect for FAIR (i.e. Findable, Accessible, Interoperable, Reusable) analyses.

In summary, our dedicated NF-Core pipeline offers a comprehensive solution for genomic imputation, covering phasing, target file imputation, and imputation quality validation stages. Using this pipeline streamlines researchers' genomic investigations, harnessing innovative tools to ensure dependable imputation outcomes across various model and non-model species. Furthermore, by adhering to FAIR principles, they contribute to standardised, reproducible, and community-supported genomic analyses.

Introduction

Sparse genetic data pose both opportunities and obstacles in genomic research. It offers a cost-effective means to study large populations, facilitating broader investigations into genetic diversity and disease susceptibility [1,2]. Genotyping micro-array has historically been the primary tool for large genome-wide association studies. While the number of variants study through those micro-array are limited (i.e. from 240k to 4M variants [3]) the confidence in the genotype observed is high. On the other hand low-coverage Whole Genome Sequencing has recently emerged as a viable low-cost alternative [4,5]. However, the increase in the number of variants observed comes at the cost of a reduced confidence in their genotypes. A low resolution as well as poor genotypes quality can hinder accurate analysis, particularly for rare variants or to elucidate complex genetic mechanisms [6].

Phasing and imputation techniques emerge as essential strategies to overcome these challenges. Phasing involves determining the parental origin of alleles, reconstructing the haplotypes present in an individual's genome. It allows to accurately predict missing genotypes (i.e. imputation) by leveraging the correlation patterns between known and unknown variants within haplotypes. Phasing can be done *de novo* with high coverage data or using an already phased panel of variants from multiple individuals as a reference [7]. However, access to a phased panel is not always possible when working with non-model species and the phasing and pre-processing steps are challenging due to the absence of clear best practices. Variability in data quality and format, coupled with numerous phasing algorithms and parameters, complicates the use of phasing techniques. Additionally, the lack of standardised guidelines for quality control measures and the need for scalability in handling large-scale datasets add further complexity to this step.

Imputation leverages known haplotype information to predict missing genotypes, effectively filling in the gaps in sparse genetic datasets. These techniques, often based on probabilistic approaches using Hidden Markov Models [8], provide likelihood scores for imputed variants, necessitating consideration of these posterior probabilities in subsequent analyses [6]. To address the usage limitation of genotype likelihood, researchers can adopt a hybrid approach, where a subset of individuals is sequenced at high coverage to ensure reliable genotype data for specific analyses. Meanwhile, the remaining individuals can be sequenced at low coverage, providing supplementary data for comprehensive analysis. This strategy allows for the acquisition of both high-quality genotype information and broader coverage across the sample population.

Implementing phasing and imputation workflows presents unique complexities. Researchers face the challenge of navigating multiple software tools, each with its own dependencies and compatibility requirements. Popular phasing tools include “Shapeit5” [9] and “PHASE 2” [10] but also “wphase” [11], “HAP2” [12], “PL-EM/triple” [13] or “Eagle2” [14]. Imputation software use will depend on the type of input data. For genotyping micro-array data, the Beagle5 [15] software is widely used by the community as well as “Impute5” [16] and “Minimac4” [17], whereas for low-coverage whole genome sequence, data is process with an available phased panel with: “Beagle5” [15], “Quilt” [18], “Glimpse2”[19]; or without one with : “Stitch”[20]. This choice complexity added to an absence of standardised environments complicate the installation, integration, and reproducibility of analysis pipelines. Moreover, some of these pieces of software are not user-friendly and the lack of documentation, the high computational loads, the susceptibility to batch artefacts and the limited availability of phased panel [6] hinder the adoption of phasing and imputation techniques in genomic research, highlighting the need for robust, user-friendly solutions.

Nextflow, a workflow language implemented in Groovy designed for scalable and reproducible scientific workflows [21], emerges as a promising solution to address these challenges. It uses a dataflow programming model that connects process nodes through channels allowing to easily parallelise huge data across chromosome or individuals. With built-in support for containerisation technologies like Docker, Conda and Singularity, Nextflow streamlines the development and deployment of complex genomic workflows. Its compatibility with popular cloud computing platforms allows researchers to leverage scalable computing resources for large-scale genomic analyses. However, some limitations have been pointed out, such as the learning curve and obscure error messages for users not familiar with dataflow programming language. Additionally, the absence of a central repository of tools, reference vocabulary, easy visualisation of the data flow, and GUI has been described as major limitations [22]. To overcome these drawbacks, the NF-Core community has made significant progress in recent years [23].

Indeed, the NF-Core community plays a pivotal role in advancing genomic analyses through standardised tools, processes and workflows written in Nextflow through collaborative efforts. By adhering to NF-Core guidelines, researchers benefit from validated, versioned, and community-supported pipelines, ensuring the reliability and reproducibility of genomic analyses [23]. To date, more than 50 high standard pipelines have been developed, most focusing on genetics, but with an increasing number being developed for other scientific fields. We can cite the most famous one nf-core/rnaseq “to analyse RNA sequencing data” [24] and nf-core/sarek “to detect variants on whole genome or targeted sequencing data” [25].

Considering these factors, the development of a dedicated NF-Core phasing and imputation pipeline, tailored to the unique demands of genomic analyses, became increasingly important. Such a pipeline would integrate state-of-the-art phasing and imputation algorithms, provide seamless compatibility with existing genomic datasets, and adhere to FAIR (i.e. Findable, Accessible, Interoperable, Reusable) principles to promote transparency and reproducibility in genomic research.

Materials and Methods

Creation

NF-Core/PhaseImpute is a Nextflow pipeline that has been developed using the NF-Core tools [23]. The pipeline is constructed following the template of NF-Core, specifically version 2.13.1, which by the means of the python « nf-core » tools streamline folder organisation, file creation, and configuration, as well as facilitates Continuous Integration workflows for Git. This pipeline, initially empty, is written in DSL2, an updated version of the Nextflow language that enhances modularisation processes. Subsequently, modules, subworkflows, and the final workflow are iteratively added. NF-Core guidelines advocate for a single tool per module, thereby enhancing code creation, updating, versioning, and sharing. Interconnections between modules are facilitated through subworkflows, which can be aggregated into a cohesive workflow structure. All sub-tools of Glimpse (version 1 and 2) and ShapeIt (version 5) have been integrated to the NF-Core repositories. Corresponding subworkflows using these tools are readily available for utilisation. Moreover, all tools are encapsulated within bio (Conda) environment and have their singularity and docker counterparts. This ensures a highly reproducible environment, guaranteeing consistent outputs for a given set of tools and data. The metro-maps presented below all have been generated with the draw.io software [26].

Data set

To assess the performance and conduct unit tests for all components of the pipeline, a dedicated data set repository has been established using data from the 1000 Genomes Project [27]. Specifically, the "1000G_2504_high_coverage" phased panel containing SNV and INDEL call sets from 3,202 high-coverage samples sourced from Byrska-Bishop et al [28] has been used. They are accessible at: http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data_collections/1000G_2504_high_coverage/working/20201028_3202_phased/. Subsequently, subregions of chr21 and chr22 between 16,57Mb-16,61 Mb have been selected from this panel, hence creating a lightweight data set of variants and significantly reducing computing time during pipeline development and testing. Both regions have been chosen as there are the ones used for testing by the GATK software and the NF-Core community. Three individuals, including one used in the Glimpse1 tutorial [29] and two other individuals, were selected for testing, namely NA12878, NA19401 and NA20359.

The distance matrix which reflects the genetic distance between pairs of individuals has been computed based on identity by descent. This was done for all 3230 individuals within the panel using PLINK (version 1.90b6.21) [30] with the --genome option, with variants filtered out when harbouring more than 20% missing genotypes (--geno 0.02) and a minor allele frequency below 1% (--maf 0.01). This allows to quantify the genetic similarity between pairs of individuals based on the proportion of their genome inherited from common ancestors. Individuals NA12878 and NA19401, each have 2 related individuals with an estimate kinship value greater than 0.2 inside the reference panel, while NA20359 has none. To ensure accurate evaluation of imputation performance, these four related individuals have been excluded from the reference panel along with the three selected individuals.

The high-coverage Compressed Read Alignment Map file (CRAM) of the three selected individuals has been retrieved and the corresponding subregions have been extracted. Genotypes has been computed from these individual files using the "mpileup" and "call" tools of "bcftools" (version 1.17) [31], serving as a validation file. This step reproduces what is done in the Glimpse tutorial. Additionally, the original CRAM files have been down-sampled to 1X to simulate low-pass data using the subsample option of the view command of "samtools" (version 1.17) [31]. Furthermore, genotypes from the SNP positions used in the Affymetrix microarray, encompassing 1,837k SNPs and Copy Number variants, have been extracted using PLINK from the called variants to simulate genotyping micro-array data.

Computational environment

All evaluations were conducted on a shared High-Performance Computing (HPC) cluster using a Slurm Scheduler on the Genouest bioinformatic platform (www.genouest.org). The computational environment was configured using mamba (version 1.5.6) and the environment specification file is available in the pipeline repository. This environment file includes openjdk (version 17.0 or higher), nextflow (version 23.10 or higher), singularity (version 3.8 or higher), nf-core (version 2.13 or higher), prettier (version 3.0 or higher) and nf-test (version 0.8 or higher) ensuring consistency and reproducibility across users. All softwares versions used during computation in the pipeline are made available in an "MultiQC" [32] html file.

Results

An overview of the pipeline is shown in Fig 1. The input data comprise a sample sheet in comma-separated value (CSV) format. This file contains the names of the target individuals as well as the paths to their genetic files (e.g., VCF, BCF, BAM, CRAM) and corresponding indices (e.g., TBI, CSI, BAI, CRAI). The pipeline features multiple entry-points, enabling various types of analyses, each generating a CSV sample sheet for subsequent analyses. The pipeline encompasses four steps: reference panel phasing and pre-processing, imputation of the target files, simulation of target files and finally concordance analysis between ground truth and imputed files. Each step can be executed individually by appending the following argument to the command line "--steps <panel_prep/impute/ simulate/validate>". The tools can be chosen in a similar way with "—tools <glimpse1/glimpse2/stitch/quilt>". The pipeline uses a reference genome given by the user and is therefore agnostic to the species study. The only limitation is that the data need to be diploid.

Each tool run by the pipeline can be easily configured by adding external arguments through ".config" files. For example, the imputation can be fine tune by adding "`--ne 100`" to the GLIMPSE1_phase process to explicit the effective diploid population size or by adding "`--window-size 200000`" to the GLIMPSE1_chunk process to increase or decrease the window-size. The more impactful parameters are also available as pipeline level parameter (i.e. prefix with "--") and can easily be adjusted from the command line such as for example: "nextflow run nf-core/phaseimpute -profile test_stitch,singularity --k_val 3 --seed 3 --outdir results" to change the Stitch software main parameters.

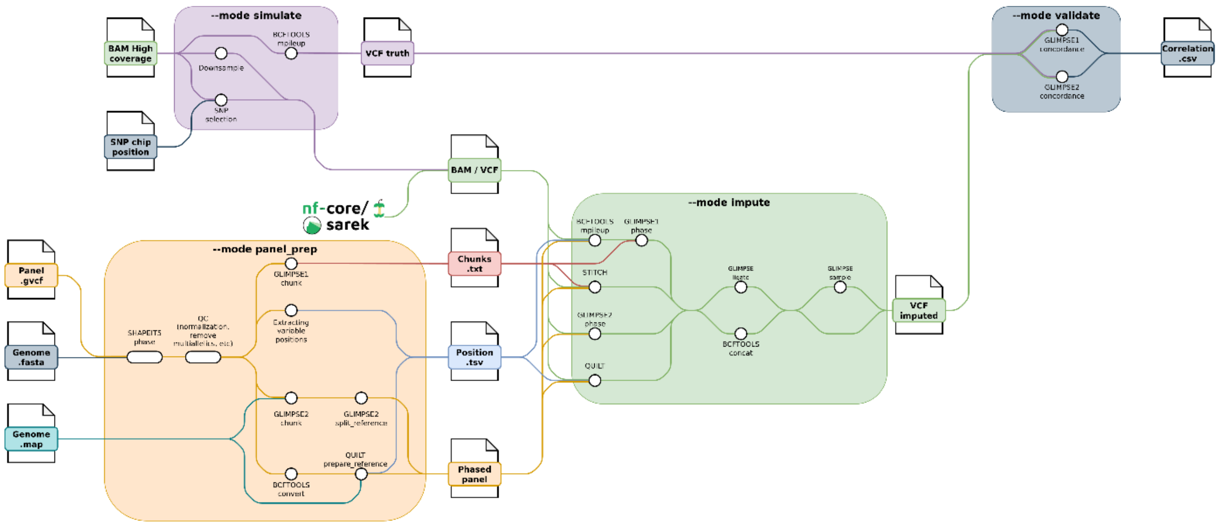


Fig 1: Overview of the "PhaseImpute" pipeline. One box per step of the pipeline (orange: pre-processing of the panel, green: target files imputation, purple: simulation of low coverage WGS and genotyping array data, grey: validation of imputed data)

Reference panel preparation and phasing:

The first step, named "panel_prep" (Fig 2) aims to compute the preprocessing required to prepare the reference panel to be used for subsequent analyses. This step involves phasing the genotypes of Genomic Variant Calling Format (GVCF) files into haplotypes and is performed using the Shapeit5 tools "phase_common" and "ligate" using a sliding window generated using the "makewindows" tool from bedtools [33]. Only "Shapeit5" is for the moment available as it is commonly used by the community and by the 1000 Genome Project. For this task, users need to also provide the reference genome of the studied species and optionally a genetic map to better determine the recombination rate along the chromosomes. Next, the procedure involves conducting quality control measures. The phased panel is first normalised and only the biallelic Single Nucleotide Polymorphisms (SNPs) are kept. These quality control steps are inspired by the Glimpse 1 tutorial [29]. Depending on the tools selected by the user, different secondary files are generated, which are necessary for the subsequent analyses.

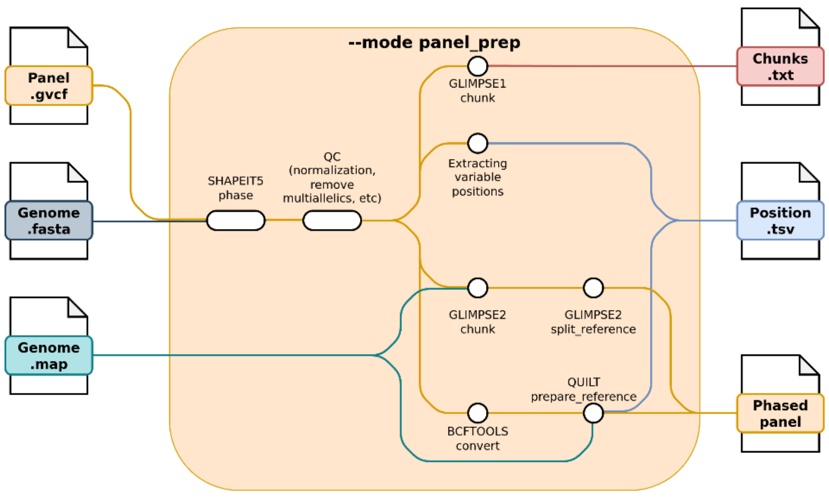


Fig 2: Metro map of the pre-processing step of the panel. This step phases a reference panel (panel .gvcf file), extracts the positions (from the Genome .fasta file) and chunks it using a genetic map (Genome .map file) for the imputation process.

Imputation of the target individuals' variants:

The second step, called "impute" (Fig 3), receives input in the form of BAM or VCF files listed in a CSV sample sheet. The data can either come from WGS (BAM or VCF format) or from genotyping array (VCF format). Variants are imputed using the different files produced in the pre-processing step and pre-existing developed subworkflows accessible within the NF-Core community. Specifically, modules and subworkflows for software tools such as Glimpse 1 and 2, Stitch and Quilt are readily available. The data by chromosomes are then aggregated and the imputed vcf is the returned.

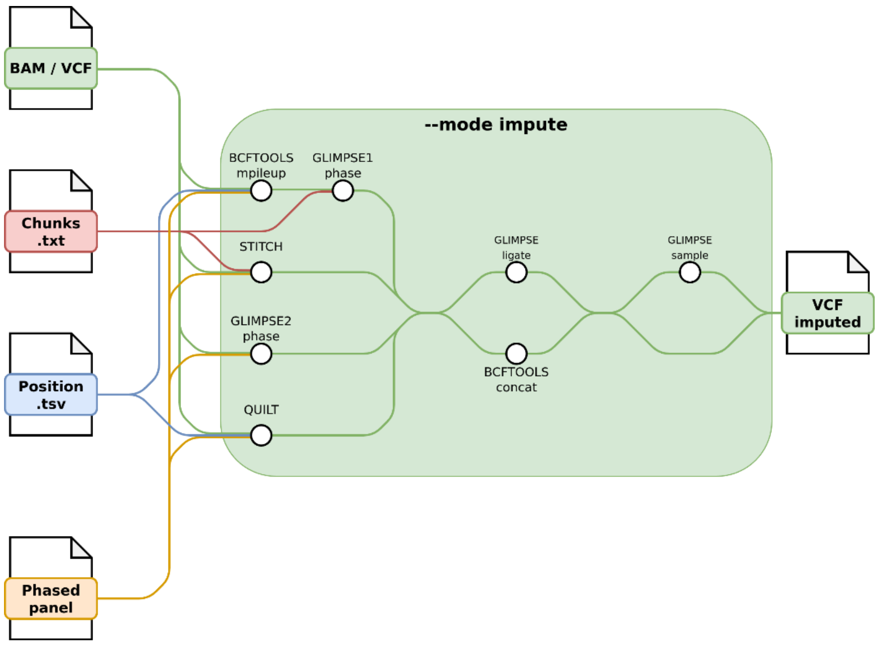


Fig 3: Metro map of the target file imputation step. This step imputes, with different softwares selected by the user, the target BAM or VCF files using the output files of the panel pre-processing and concatenate the chunks for each target individuals. The phased reference panel is optional for the Stitch software.

Simulation and concordance analysis

Finally, the "simulate" and "validate" steps respectively enable the generation of sparse genetic data from high-coverage whole genome sequence data and the comparison of imputed data with their original counterpart. The simulation process, depicted in Fig 4, facilitates either down sampling files to a specified mean coverage from high coverage sequencing or selecting specific SNP chip array positions to simulate SNP chip data. The original high coverage WGS data are also processed with the "mpileup" tool to create a ground "truth" variants file.

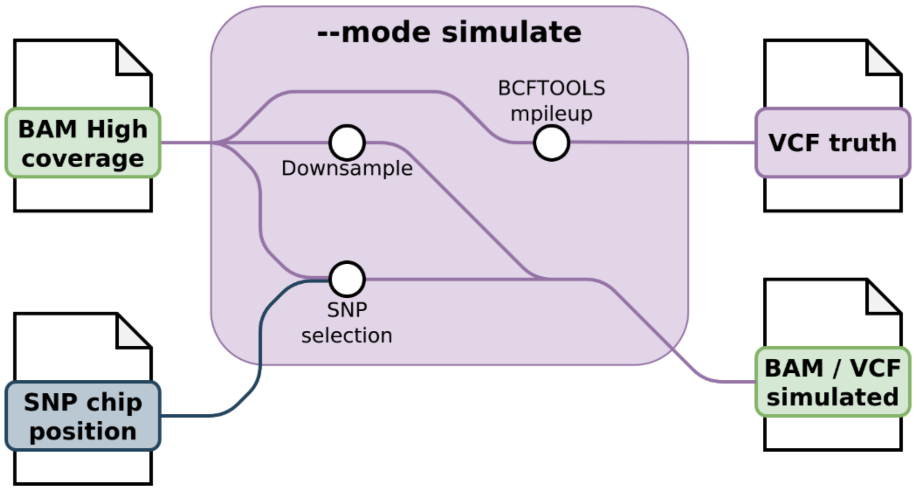


Fig 4: Metro map of the simulation step. This step allows the creation of simulated low coverageWGS or genotyping array data from high coverage WGS (more than 20X). The SNP chip position file is optional if the user only wants to simulate low coverage WGS data. The variants from the full data are also computed using the bcftools "mpileup" process.

The "validate" step (Fig 5), conversely, uses the Glimpse 1 and 2 concordance tools to perform statistical analysis. This step requires input from both the ground truth and the imputed data from the simulated data. It then output a csv file recording the matching genotype rate.

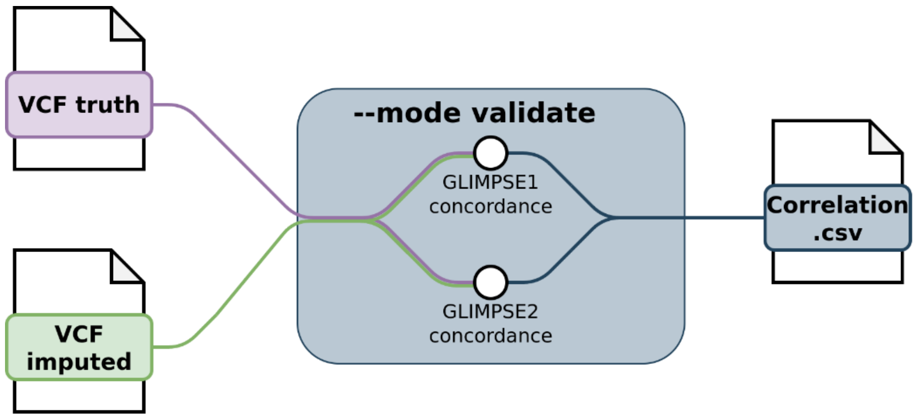


Fig 5: Metro map of the validation step. A vcf file considered as the truth is here compared to the vcf file imputed by the pipeline using the "concordance" tool of Glimpse.

Unit testing

Data set

The data set used for unit-testing comprises a reference panel phased variants files consisting of 860 variants on the chromosome 21 and 918 variants for the chromosome 22 for a total of 3,195 individuals. The reference genome employed is the Genome Reference Consortium Human Build 38 (GRCh38). Table 1 presents the statistical information obtained by the "coverage" tool of samtools for the three individuals selected as targets for the test dataset.

Individuals	Chr	numreads	covbases	coverage	meandepth	meanbaseq	meanmapq
NA12878	21	8475	40001	100	31.5299	29.6	59.7
	22	8973	40001	100	33.1928	29.5	57.3
NA20359	21	8143	40000	99.9975	30.1806	29.3	59.8
	22	8384	40001	100	30.9138	29.1	57.3
NA19401	21	8741	39996	99.9875	32.4854	29.6	59.8
	22	8682	40001	100	32.0736	29.5	57.7

Table 1: Summary statistics of the three individuals selected for the region chr21: 16,57-16,61 Mb and chr22: 16,57-16,61 Mb. numreads: Number reads aligned to the region (after filtering), covbases: Number of covered bases with depth >= 1, coverage: Percentage of covered bases [0..100], meandepth: Mean depth of coverage, meanbaseq: Mean baseQ in covered region, meanmapq: Mean mapQ of selected reads.

Modules and subworkflows

The entire NF-Core community has recently opted to transition all their python test procedures to Groovy-based test procedures available through the NF-test plugin. Each module and subworkflow undergoes separate testing using the "PhaseImpute" test-dataset to ensure that each component operates as intended. This new unit test workflow is structured such that each module includes all the files that are needed for its execution and testing within its own folder. This organisational approach improves visualisation and code debugging. A module is therefore organised as depicted in Fig 6.

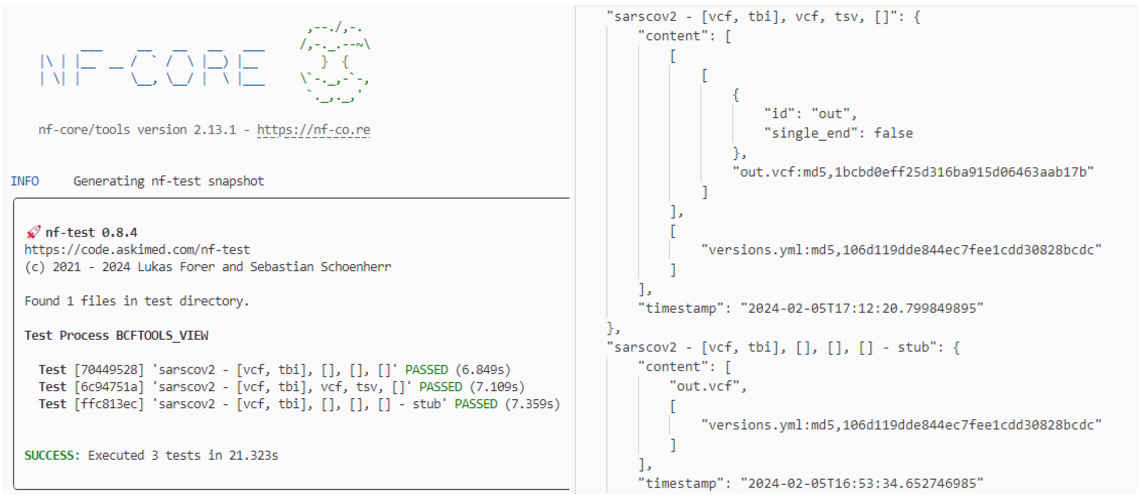
```
modules / subworkflows:
    - local:
    - nf-core:
        - tool:
            - subtool:
                - tests:
                    - main.nf.test      # Process code for the unit tests
                    - main.nf.test.snap # Snapshot of the expected results
                    - nextflow.config   # Configuration for the unit tests
                    - tags.yml          # Tags and path for the unit test
                - environment.yml       # Environment file of the tool
                - main.nf               # Process code
                - meta.yml              # Documentation describing the tool
                - subtools.diff         # Git diff text to patch modification
```

Fig 6: NF-Core file organisation for modules and subworkflows with NF-test

The advantage of such an organisation is the ability to directly test a module or a subworkflow with the simple following command:

```
nf-core <modules/subworflows> test tool/subtool
--profile <singularity/docker/conda>
```

This command will run the "main.nf.test" process and create a snapshot of the results if none are available. The test will be run a second time to check its stability. An example of such an output can be seen in Fig 7.



```
    nf-core/tools version 2.13.1 - https://nf-co.re


INFO     Generating nf-test snapshot

 nf-test 0.8.4
 https://code.askimed.com/nf-test
 (c) 2021 - 2024 Lukas Forer and Sebastian Schoenherr

 Found 1 files in test directory.

 Test Process BCFTOOLS_VIEW

   Test [70449528] 'sarscov2 - [vcf, tbi], [], [], []' PASSED (6.849s)
   Test [6c94751a] 'sarscov2 - [vcf, tbi], vcf, tsv, []' PASSED (7.109s)
   Test [ffc813ec] 'sarscov2 - [vcf, tbi], [], [], [] - stub' PASSED (7.359s)


 SUCCESS: Executed 3 tests in 21.323s
```

```
    "sarscov2 - [vcf, tbi], vcf, tsv, []": {
        "content": [
            [
                [
                    {
                        "id": "out",
                        "single_end": false
                    },
                    "out.vcf:md5,1bcbd0eff25d316ba915d06463aab17b"
                ]
            ],
            [
                "versions.yml:md5,106d119dde844ec7fee1cdd30828bcdc"
            ]
        ],
        "timestamp": "2024-02-05T17:12:20.799849895"
    },
    "sarscov2 - [vcf, tbi], [], [], [] - stub": {
        "content": [
            "out.vcf",
            [
                "versions.yml:md5,106d119dde844ec7fee1cdd30828bcdc"
            ]
        ],
        "timestamp": "2024-02-05T16:53:34.652746985"
```

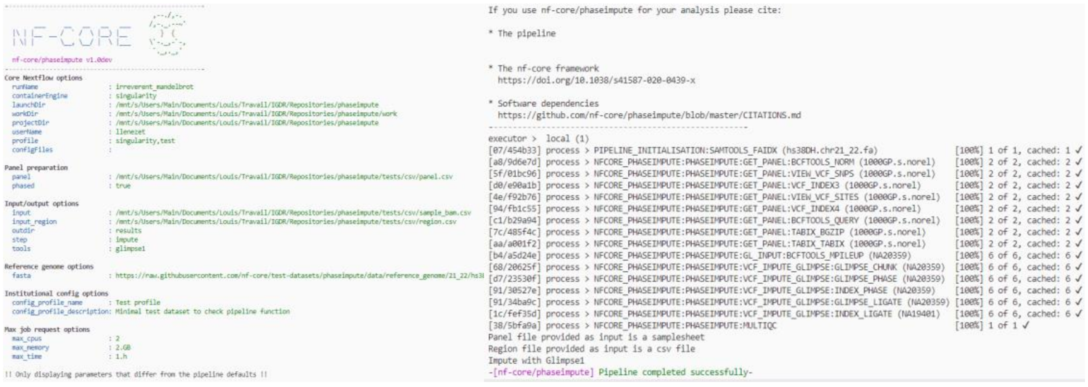
Fig 7: Output of nf-test tool for a nf-core module (bcftools / view) on the left and a section of the snapshot generated through the test on the right.

Pipeline

The pipeline also features a test workflow that can be easily initiated using:

```
nextflow run nf-core/phaseimpute --profile test,<singularity/docker/conda>
--outdir results
```

This test uses the “PhaseImpute” test data set and currently executes the panel preparation step, followed by the imputation of target files for the three selected individuals using the tool Glimpse 1 for the subregions of chromosomes 21 and 22. The output of this test can be seen in Fig 8.



```
NF-CORE
nf-core/phaseimpute v1.0dev
-----------------------------------------------------
Core Nextflow options
  runName                   : irreverent_mandelbrot
  containerEngine           : singularity
  launchDir                 : /mnt/s/Users/Main/Documents/Louis/Travail/IGDR/Repositories/phaseimpute
  workDir                   : /mnt/s/Users/Main/Documents/Louis/Travail/IGDR/Repositories/phaseimpute/work
  projectDir                : /mnt/s/Users/Main/Documents/Louis/Travail/IGDR/Repositories/phaseimpute
  userName                  : llenezet
  profile                   : singularity,test
  configFiles               :

Panel preparation
  panel                     : /mnt/s/Users/Main/Documents/Louis/Travail/IGDR/Repositories/phaseimpute/tests/csv/panel.csv
  phased                    : true

Input/output options
  input                     : /mnt/s/Users/Main/Documents/Louis/Travail/IGDR/Repositories/phaseimpute/tests/csv/sample_bam.csv
  input_region              : /mnt/s/Users/Main/Documents/Louis/Travail/IGDR/Repositories/phaseimpute/tests/csv/region.csv
  outdir                    : results
  step                      : impute
  tools                     : glimpse1

Reference genome options
  fasta                     : https://raw.githubusercontent.com/nf-core/test-datasets/phaseimpute/data/reference_genome/21_22/hs38

Institutional config options
  config_profile_name       : Test profile
  config_profile_description: Minimal test dataset to check pipeline function

Max job request options
  max_cpus                  : 2
  max_memory                : 2.GB
  max_time                  : 1.h

!! Only displaying parameters that differ from the pipeline defaults !!
```

```
If you use nf-core/phaseimpute for your analysis please cite:

* The pipeline


* The nf-core framework
  https://doi.org/10.1038/s41587-020-0439-x

* Software dependencies
  https://github.com/nf-core/phaseimpute/blob/master/CITATIONS.md
-----------------------------------------------------
executor >  local (1)
[07/454b33] process > PIPELINE_INITIALISATION:SAMTOOLS_FAIDX (hs38DH.chr21_22.fa)                      [100%] 1 of 1, cached: 1 ✓
[a8/9d6e7d] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:GET_PANEL:BCFTOOLS_NORM (1000GP.s.norel)          [100%] 2 of 2, cached: 2 ✓
[5f/01bc96] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:GET_PANEL:VIEW_VCF_SNPS (1000GP.s.norel)          [100%] 2 of 2, cached: 2 ✓
[d0/e90a1b] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:GET_PANEL:VCF_INDEX3 (1000GP.s.norel)             [100%] 2 of 2, cached: 2 ✓
[4e/f92b76] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:GET_PANEL:VIEW_VCF_SITES (1000GP.s.norel)         [100%] 2 of 2, cached: 2 ✓
[94/fb1c55] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:GET_PANEL:VCF_INDEX4 (1000GP.s.norel)             [100%] 2 of 2, cached: 2 ✓
[c1/b29a94] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:GET_PANEL:BCFTOOLS_QUERY (1000GP.s.norel)         [100%] 2 of 2, cached: 2 ✓
[7c/485f4c] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:GET_PANEL:TABIX_BGZIP (1000GP.s.norel)            [100%] 2 of 2, cached: 2 ✓
[aa/a001f2] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:GET_PANEL:TABIX_TABIX (1000GP.s.norel)            [100%] 2 of 2, cached: 2 ✓
[b4/a5d24e] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:GL_INPUT:BCFTOOLS_MPILEUP (NA20359)               [100%] 6 of 6, cached: 6 ✓
[68/20625f] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:VCF_IMPUTE_GLIMPSE:GLIMPSE_CHUNK (NA20359)        [100%] 6 of 6, cached: 6 ✓
[d7/23530f] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:VCF_IMPUTE_GLIMPSE:GLIMPSE_PHASE (NA20359)        [100%] 6 of 6, cached: 6 ✓
[91/30527e] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:VCF_IMPUTE_GLIMPSE:INDEX_PHASE (NA20359)          [100%] 6 of 6, cached: 6 ✓
[91/34ba9c] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:VCF_IMPUTE_GLIMPSE:GLIMPSE_LIGATE (NA20359)       [100%] 6 of 6, cached: 6 ✓
[1c/fef35d] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:VCF_IMPUTE_GLIMPSE:INDEX_LIGATE (NA19401)         [100%] 6 of 6, cached: 6 ✓
[38/5bfa9a] process > NFCORE_PHASEIMPUTE:PHASEIMPUTE:MULTIQC                                           [100%] 1 of 1 ✓
Panel file provided as input is a samplesheet
Region file provided as input is a csv file
Impute with Glimpse1
-[nf-core/phaseimpute] Pipeline completed successfully-
```

Fig 8: Output of the test workflow of the “PhaseImpute” pipeline. A description of the parameters used is first prompted (on the left) followed by the different steps (on the right).

Discussion

Using a dedicated workflow language such as Nextflow or Snakemake offers significant advantages for processing high-throughput data seamlessly while ensuring high reproducibility [21,34]. The NF-Core initiative, aimed at creating highly standardised tools, modules, subworkflows, workflows and template empowers users to rapidly develop a high-quality pipeline. However, it is important to acknowledge that learning the Nextflow language and adhering to the NF-Core guidelines requires time and effort. Nonetheless, this investment allows for the creation of Findable, Accessible, Interoperable, and Reusable (FAIR) workflows that are accessible to all.

By integrating each tool into the NF-Core repository, users access the latest versions, maintaining cutting-edge workflows. Moreover, the repository's inclusion of pre-developed modules eliminates redundant development. nf-core's guidelines ensure versatile tool usage through standardised input-output architecture. The “nf-core” tool and GitHub Continuous Integration (CI) also facilitate tracking of changes, linting and automatic testing all the processes and workflows. This ensures that the pipeline remains robust and up to date. Archiving each version within Zenodo [35] enables reproducibility of analyses even decades later, ensuring proper tools versioning, without, normally, encountering any issues.

Looking ahead, the “PhaseImpute” pipeline holds promise for advancing genomic analyses further. Future efforts will focus on integrating new features, refining existing functionalities, and adapting to evolving needs of the rapidly growing field of genetic imputation. For example, a full-scale test will be available using all the full chromosomes available from the 1000 Genome Project and information about the CO2 footprint of the runs will be accessible through the integration of the nf-co2footprint plugin. Community engagement will play a crucial role in shaping the pipeline's development trajectory, fostering collaboration, and soliciting feedback from the broader bioinformatics community. Benchmarking will be essential for evaluating the pipeline's CPU and memory usage, providing insights into its efficiency and scalability across diverse computational environments. This will be done with the first stable release through the integrated tools available through NF-Core and the nf-co2footprint plugin.

To conclude, the “PhaseImpute” pipeline represents progress in genomic imputation and analysis. By providing a standardised, reproducible, and community-supported solution, it contributes to advancing genomic research. This work underscores the importance of collaborative efforts in developing effective tools and workflows for the scientific community.

Availability and implementation

The “PhaseImpute” pipeline is accessible at https://nf-co.re/phaseimpute/dev. Although it is currently under development, a stable release should be available in the coming months. The test dataset is located in the “test-datasets” repository of the NF-Core community, specifically under its dedicated branch, accessible at https://github.com/nf-core/test-datasets/tree/phaseimpute.

Acknowledgements

The authors express their gratitude to the NF-Core and Nextflow community for their efforts in developing the NF-Core infrastructure and resources for Nextflow pipelines. A comprehensive list of NF-Core community members can be found at https://nf-co.re/community. More particularly the authors deeply thank Maxime Garcia, Anabella Trigila and Saul Pierotti for their contribution to the pipeline and their reviews.

The authors also thank Anthony Herzig and Thomas Derrien for their careful review of this paper.

The authors extend theirs thanks to the GenOuest Bioinformatics of the University of Rennes for generously providing the high-performance computing infrastructure.

Funding information

This work has been made possible by the funding of the Visio foundation and the partnership of the guide dogs association: “Association les Chiens Guides d’Aveugles de l’Ouest”.

References

1. Li JH, Mazur CA, Berisa T, Pickrell JK. Low-pass sequencing increases the power of GWAS and decreases measurement error of polygenic risk scores compared to genotyping arrays. Genome Res. 2021;31:529-37.

2. Mazzonetto PC, Villela D, da Costa SS, Krepischi ACV, Milanezi F, Migliavacca MP, et al. Low-pass whole genome sequencing is a reliable and cost-effective approach for copy number variant analysis in the clinical setting. Ann. Hum. Genet. 2024;88:113-25.

3. Verlouw JAM, Clemens E, de Vries JH, Zolk O, Verkerk AJMH, am Zehnhoff-Dinnesen A, et al. A comparison of genotyping arrays. Eur. J. Hum. Genet. 2021;29:1611-24.

4. Pasaniuc B, Rohland N, McLaren PJ, Garimella K, Zaitlen N, Li H, et al. Extremely low-coverage sequencing and imputation increases power for genome-wide association studies. Nat. Genet. 2012;44:631-5.

5. Chat V, Ferguson R, Morales L, Kirchhoff T. Ultra Low-Coverage Whole-Genome Sequencing as an Alternative to Genotyping Arrays in Genome-Wide Association Studies. Front. Genet. [Internet] 2022 [cité 2024 mars 15];12. Available from: https://www.frontiersin.org/journals/genetics/articles/10.3389/fgene.2021.790445

6. Lou RN, Jacobs A, Wilder AP, Therkildsen NO. A beginner's guide to low-coverage whole genome sequencing for population genomics. Mol. Ecol. 2021;30:5966-93.

7. Delaneau O, Coulonges C, Zagury JF. Shape-IT: new rapid and accurate algorithm for haplotype inference. BMC Bioinformatics 2008;9:540.

8. Marino AD, Mahmoud AA, Bose M, Bircan KO, Terpolovsky A, Bamunusinghe V, et al. A comparative analysis of current phasing and imputation software. PLOS ONE 2022;17:e0260177.

9. Hofmeister RJ, Ribeiro DM, Rubinacci S, Delaneau O. Accurate rare variant phasing of whole-genome and whole-exome sequencing data in the UK Biobank. Nat. Genet. 2023;1-7.

10. Stephens M, Scheet P. Accounting for decay of linkage disequilibrium in haplotype inference and missing-data imputation. Am. J. Hum. Genet. 2005;76:449-62.

11. Fearnhead P, Donnelly P. Estimating recombination rates from population genetic data. Genetics 2001;159:1299-318.

12. Lin S, Chakravarti A, Cutler DJ. Haplotype and missing data inference in nuclear families. Genome Res. 2004;14:1624-32.

13. Niu T, Qin ZS, Xu X, Liu JS. Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. Am. J. Hum. Genet. 2002;70:157-69.

14. Loh PR, Danecek P, Palamara PF, Fuchsberger C, A Reshef Y, K Finucane H, et al. Reference-based phasing using the Haplotype Reference Consortium panel. Nat. Genet. 2016;48:1443-8.

15. Browning BL, Zhou Y, Browning SR. A One-Penny Imputed Genome from Next-Generation Reference Panels. Am. J. Hum. Genet. 2018;103:338-48.

16. Rubinacci S, Delaneau O, Marchini J. Genotype imputation using the Positional Burrows Wheeler Transform. PLOS Genet. 2020;16:e1009049.

17. Das S, Forer L, Schönherr S, Sidore C, Locke AE, Kwong A, et al. Next-generation genotype imputation service and methods. Nat. Genet. 2016;48:1284-7.

18. Davies RW, Kucka M, Su D, Shi S, Flanagan M, Cunniff CM, et al. Rapid genotype imputation from sequence with reference panels. Nat. Genet. 2021;53:1104-11.

19. Rubinacci S, Hofmeister RJ, Sousa da Mota B, Delaneau O. Imputation of low-coverage sequencing data from 150,119 UK Biobank genomes. Nat. Genet. 2023;1-3.

20. Davies RW, Flint J, Myers S, Mott R. Rapid genotype imputation from sequence without reference panels. Nat. Genet. 2016;48:965-9.

21. Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. Nat. Biotechnol. 2017;35:316-9.

22. Cohen-Boulakia S, Belhajjame K, Collin O, Chopard J, Froidevaux C, Gaignard A, et al. Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities. Future Gener. Comput. Syst. 2017;75:284-98.

23. Ewels PA, Peltzer A, Fillinger S, Patel H, Alneberg J, Wilm A, et al. The nf-core framework for community-curated bioinformatics pipelines. Nat. Biotechnol. 2020;38:276-8.

24. Patel H, Ewels P, Peltzer A, Manning J, Botvinnik O, Sturm G, et al. nf-core/rnaseq: nf-core/rnaseq v3.14.0 - Hassium Honey Badger [Internet]. 2024 [cité 2024 mars 15];Available from: https://zenodo.org/records/10471647

25. Garcia M, Juhos S, Larsson M, Olason PI, Martin M, Eisfeldt J, et al. Sarek: A portable workflow for whole-genome sequencing analysis of germline and somatic variants [Internet]. 2020 [cité 2024 mars 15];Available from: https://f1000research.com/articles/9-63

26. Benson D. Draw.io a JavaScript, client-side editor for general diagramming. [Internet]. 2024 [cité 2024 mai 23];Available from: https://github.com/jgraph/drawio

27. 1000 Genomes Project Consortium, Auton A, Brooks LD, Durbin RM, Garrison EP, Kang HM, et al. A global reference for human genetic variation. Nature 2015;526:68-74.

28. Byrska-Bishop M, Evani US, Zhao X, Basile AO, Abel HJ, Regier AA, et al. High-coverage whole-genome sequencing of the expanded 1000 Genomes Project cohort including 602 trios. Cell 2022;185:3426-3440.e19.

29. Delaneau O, Rubinacci S. GLIMPSE tutorial [Internet]. Genotype Likelihoods Imput. PhaSing MEthod GLIMPSE Tutor. B382021 [cité 2024 mars 15];Available from: https://odelaneau.github.io/GLIMPSE/glimpse1/tutorial_b38.html

30. Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, et al. PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses. Am. J. Hum. Genet. 2007;81:559-75.

31. Danecek P, Bonfield JK, Liddle J, Marshall J, Ohan V, Pollard MO, et al. Twelve years of SAMtools and BCFtools. GigaScience 2021;10:giab008.

32. Ewels P, Magnusson M, Lundin S, Käller M. MultiQC: summarize analysis results for multiple tools and samples in a single report. Bioinformatics 2016;32:3047-8.

33. Quinlan AR, Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinforma. Oxf. Engl. 2010;26:841-2.

34. Mölder F, Jablonski KP, Letcher B, Hall MB, Tomkins-Tinch CH, Sochat V, et al. Sustainable data analysis with Snakemake [Internet]. 2021 [cité 2024 mars 15];Available from: https://f1000research.com/articles/10-33

35. Potter M, Smith T. Making code citable with Zenodo and GitHub. 2015 [cité 2024 mai 21];Available from: https://zenodo.org/records/45042