POLITECNICO DI TORINO
Repository ISTITUZIONALE

snputils: A High-Performance Python Library for Genetic Variation and Population Structure

Original
snputils: A High-Performance Python Library for Genetic Variation and Population Structure / Bonet, D., Comajoan Cara, M., Barrabés, M., Smeriglio, R., Geleta, M., Dominguez Mantes, A., Thomassin, C., Agrawal, D., Shanks, C., Huang, E.C., Aounallah, K., Franquesa Monés, M., Luis, A., Saurina, J., Perera, M., López, C., Jaras, A., Oriol Sabat, B., Abante, J., Moreno-Grau, S., et al.. - In: MOLECULAR BIOLOGY AND EVOLUTION. - ISSN 1537-1719. - (In corso di stampa).

Availability:
This version is available at: 11583/3016081 since: 2026-09-29T17:31:37Z

Publisher:
Oxford University Press

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Oxford University Press preprint/submitted version

This article has been accepted for publication in MOLECULAR BIOLOGY AND EVOLUTION Published by Oxford University Press

(Article begins on next page)

01 October 2026

Molecular Biology and Evolution, 2026, , 1–12
DOI added during production
Published: Date added during production
Resources

snputils: A High-Performance Python Library for Genetic Variation and Population Structure

David Bonet,[1,2,3,†] Marçal Comajoan Cara,[1,2,4,†] Míriam Barrabés,[1,5,†] Riccardo Smeriglio,[6] Margarita Geleta,[1,4] Albert Dominguez Mantes,[7] Christophe Thomassin,[1,8] Devang Agrawal,[9] Cole Shanks,[2] Edward C. Huang,[10] Khaled Aounallah,[9] Marc Franquesa Monés,[11] Aina Luis,[11] Joan Saurina,[11] Maria Perera,[5] Cayetana López,[11] Ainel Jaras,[1] Benet Oriol Sabat,[12] Jordi Abante,[3,13] Sonia Moreno-Grau,[1] Daniel Mas Montserrat[1] and Alexander G. Ioannidis[1,2,9,*]

[1]Department of Biomedical Data Science, Stanford University, Stanford, CA, USA
[2]Genomics Institute, University of California, Santa Cruz, Santa Cruz, CA, USA
[3]Department of Biomedical Sciences, Universitat de Barcelona, Barcelona, Catalonia, Spain
[4]Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA, USA
[5]Munster Technological University, Cork, Ireland
[6]Politecnico di Torino, Turin, Italy
[7]School of Life Sciences, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
[8]École Polytechnique, Palaiseau, France
[9]Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA, USA
[10]Department of Computer Science, Stanford University, Stanford, CA, USA
[11]Department of Signal Theory and Communications, Universitat Politècnica de Catalunya, Barcelona, Catalonia, Spain
[12]Department of Computer Science, University of California, Los Angeles, Los Angeles, CA, USA
[13]Institute of Neurosciences (UBNeuro), Universitat de Barcelona, Barcelona, Catalonia, Spain
*Correspondence: ioannidis@stanford.edu. †These authors contributed equally.

Abstract

Large-scale genomic and population-genetic datasets offer unprecedented opportunities to study population structure and uncover the genetic basis of complex traits and diseases. Existing analytical tools, however, are characterized by format incompatibilities, limited functionality, and computational inefficiencies, forcing researchers to construct pipelines chaining together fragmented command-line utilities and ad hoc scripts. These are difficult to maintain, scale, and reproduce. We present `snputils`, an open-source Python library for high-performance processing and analysis of genotype, ancestry, phenotype, and identity-by-descent data within a single framework suitable for biobank-scale research. `snputils` provides unified data containers and efficient routines for genotype quality control, filtering, merging, and computing classical population-genetic statistics, with optional ancestry-specific masking. An identity-by-descent module supports reading multiple formats, filtering, and ancestry-restricted segment trimming for relatedness and demographic inference. `snputils` also incorporates ancestry masking and multi-array functionalities for dimensionality reduction methods, as well as efficient implementations of admixture simulation, admixture mapping, genome-wide association testing, and advanced visualization capabilities. With support for the most commonly used file formats, `snputils` natively supports human and non-human diploid datasets represented in standard formats. At the same time, its modular and optimized design reduces technical overhead, facilitating reproducible workflows that accelerate discoveries in population genetics, genomic research, and precision medicine. Benchmarks show faster I/O and concordant PCA, allele-frequency statistics, and admixture mapping results relative to established reference tools, with substantial runtime reductions. `snputils` is available at https://github.com/AI-sandbox/snputils, with documentation and tutorials at docs.snputils.org.

Keywords population genetics, computational genomics, bioinformatics, ancestry, identity-by-descent, open-source software

Introduction

The rapid expansion of large-scale genomic datasets, driven by continuing advances in affordable high-throughput sequencing technologies, has enabled major breakthroughs in genomics and is reshaping biomedical discovery and clinical translation (Gaziano et al., 2016; Nagai et al., 2017; Bycroft et al., 2018; Feng et al., 2022; Kurki et al., 2023; All of Us Research Program Genomics Investigators, 2024; Cook et al., 2025; Barberena-Jonas et al., 2026). Modern human biobanks now profile millions of variants across hundreds of thousands to millions

Received: Date Month Year. **Revised received:** Date Month Year. **Accepted:** Date Month Year
© The Author(s) 2026. Copyright and licence statements to be updated by the publisher during production.

of individuals. These datasets have accelerated genome-wide association studies (Uffelmann et al., 2021; Abdellaoui et al., 2023; Dias et al., 2025), polygenic risk score development (Chatterjee et al., 2016; Khera et al., 2018; Phulka et al., 2023; Moreno-Grau et al., 2024), ancestry inference (Alexander et al., 2009; Hilmarsson et al., 2026; Dominguez Mantes et al., 2023; Saurina-i Ricos et al., 2026), genotype simulation (Baumdicker et al., 2022; Perera et al., 2022; Haller et al., 2026; Geleta et al., 2026b), demographic reconstruction (Browning and Browning, 2015; Ioannidis et al., 2020), and clinical trait mapping (Tanigawa et al., 2019; COVID-19 Host Genetics Initiative, 2021; Karczewski et al., 2022). Biobanks support the identification of disease mechanisms, prioritization of therapeutic targets, and creation of risk stratification strategies that can inform screening and prevention across diverse populations. As these biobanks become increasingly representative of global genetic diversity, scalable packages for analyzing population genetic variation are becoming crucial. Yet reproducible analysis at this scale remains limited by persistent computational bottlenecks, heterogeneous file formats, and an ecosystem of tools with narrow scopes and limited interoperability.

In practice, researchers often assemble pipelines from multiple tools with incompatible input and output formats. As a result, even standard workflows frequently require repeated format conversion, intermediate files, and custom code to move between genotype processing, ancestry inference, association testing, visualization, and downstream statistical analysis. This fragmentation increases maintenance burden and can introduce failures or silent data corruption during manual format conversion. As biobank-scale studies grow in size and complexity, there is a critical need for unified, robust, and adaptable computational tools that can alleviate these bottlenecks and sustain the pace of genomic discovery in diverse cohorts. A range of existing tools addresses individual components of genomic workflows, each with distinct trade-offs in scalability, flexibility, and analytical breadth (Purcell et al., 2007; Danecek et al., 2011; Yang et al., 2011; Patterson et al., 2012; Chang et al., 2015; Kadie and Heckerman, 2017; Pedersen and Quinlan, 2017; Miles et al., 2020; Danecek et al., 2021; Maier et al., 2023; sgkit developers, 2021; Poterba et al., 2025).

Command-line tools remain central to large-scale processing. PLINK 1/2 (Purcell et al., 2007; Chang et al., 2015) provide highly efficient implementations for genome-wide association studies (GWAS) and population-genetic analyses, while BCFtools (Danecek et al., 2021) and VCFtools (Danecek et al., 2011) offer widely adopted utilities for manipulating, filtering, validating, and summarizing variant files. EIGENSOFT remains a standard reference for population-structure inference and stratification correction through `smartpca` and EIGENSTRAT-style association testing (Patterson et al., 2006). These tools are mature, but their core workflow is file- and command-line-centric, which makes interactive, exploratory, and composable analyses less straightforward than in notebook-native environments. Several packages extend this ecosystem with more specialized statistical methods. GCTA (Yang et al., 2011) adds complementary functionality for genetic relationship matrix construction, variance-component and heritability estimation, and mixed-model association tests that scale efficiently to large cohorts. ADMIXTOOLS 1/2 (Patterson et al., 2012; Maier et al., 2023) remains widely used for allele-frequency-correlation methods, including f-statistics, qpWave, qpAdm, and qpGraph. Hail (Poterba et al., 2025) leverages distributed computing via Spark and offers an expressive language for combining multiple steps, including dataset merging, quality control, and statistical modeling, into a single pipeline. Existing Python-native libraries for genomics are also specialized. For instance, scikit-allel (Miles et al., 2020) supports flexible exploratory analyses using NumPy arrays and Pandas integration. sgkit (sgkit developers, 2021) builds on Dask and xarray, providing a scalable framework for quantitative and population-genetic workflows, including modules for genomic ancestry-proportion inference. Other libraries include pgenlib (Chang et al., 2015), cyvcf2 (Pedersen and Quinlan, 2017) and pysnptools (Kadie and Heckerman, 2017), which provide Pythonic data access but limited built-in analysis.

We developed `snputils`, a Python library that integrates high-performance I/O, data transformation, querying, analysis, simulation, and advanced visualization within a unified framework. The library supports a broad range of widely used genotype, ancestry, phenotype, and identity-by-descent (IBD) formats, as well as ancestry-aware workflows and performance gains through optimized I/O, vectorized computation, and optional GPU acceleration. Previously, users faced a trade-off: command-line tools offer efficiency but lack Python integration and flexibility, while Python-based alternatives vary in scalability and feature coverage. `snputils` addresses these limitations by combining the computational efficiency of traditional command-line tools with the programmatic flexibility of Python-based frameworks, while providing integrated ancestry and phenotype analysis capabilities. The library supports chunked processing and on-demand data loading, enabling analysis of large-scale genomic datasets that may exceed available RAM. Core algorithms are implemented using vectorized operations with NumPy (Harris et al., 2020) and optimized routines, while I/O optimization strategies enable rapid access to specific genomic regions, sample subsets, or data fields without full dataset materialization. Additionally, the library incorporates optional GPU acceleration for matrix operations and data transformations, delivering substantial speedups for memory-intensive genomic computations.

Although motivated by human biobank analysis, the core `snputils` data model is organism-agnostic for diploid datasets represented in standard variant, sample, phenotype, and population-label formats. Consequently, core workflows for genotype I/O, filtering, PCA, allele-frequency aggregation, population-genetic statistics, association testing, and simulation can also be applied to non-human diploid datasets when appropriate metadata and coordinate systems are provided. Human-oriented utilities, such as chromosome-label normalization and chromosome-painting defaults, are implemented as optional helpers rather than assumptions of the core architecture.

Table 1 summarizes the key features of `snputils`, and Table 2 compares `snputils` with established genotype and population-genetic toolkits across major computational and analytical capabilities. A compact quickstart example showing how the same interface links I/O, conversion, ancestry-aware summaries, visualization, GWAS, and admixture mapping is shown in Listing 1.

Table 1. Overview of the `snputils` analysis framework. `snputils` organizes genotype, ancestry, phenotype, identity-by-descent, and genomic-interval data into aligned Python objects that can be filtered, analyzed, visualized, and exported through a shared interface. The framework combines multi-format I/O including plain, gzip (`.gz`) and Zstandard (`.zst`) compressed text inputs and native binary formats, with ancestry-aware population-genetic methods, association testing, simulation, relatedness analysis, and scalable execution.

Workflow layer	Primary inputs and objects	Core capabilities	Practical role in an analysis
Data access and harmonization			
Genotype variation	VCF (`.vcf`), BCF (`.bcf`), PLINK 1 (`.bed`, `.bim`, `.fam`), PLINK 2 (`.pgen`, `.pvar`, `.psam`), BGEN (`.bgen`, `.sample`), and GRG files (`.trees`, `.grg`) loaded into aligned `SNPObject` and `GRGObject` containers.	Automatic format detection, region and sample selection, chunked iteration, chromosome-label normalization, genotype QC, allele-flip detection, strand-ambiguous filtering, merging, and direct export.	Reduces custom file-conversion code and keeps genotype matrices synchronized with variant coordinates, alleles, and sample metadata.
Ancestry and segment data	Locus-specific ancestry files in MSP (`.msp`, `.msp.tsv`), FLARE VCF (`.anc.vcf`), or LANC (`.lanc`) format; genomic ancestry-proportion Q/P matrices and labels (`.Q`, `.P`, `.txt`, `.fam`, `.bim`, `.map`); phenotype tables (`.txt`, `.phe`, `.pheno`, `.phen`, `.csv`, `.tsv`, `.map`, `.smap`, `.xlsx`); metadata and covariates; and IBD from hap-IBD (`.ibd`) or ancIBD (`.tsv` or directory).	Typed containers for locus-specific ancestry, genomic ancestry proportions, phenotype, and identity-by-descent data with shared filtering, subsetting, and metadata alignment operations.	Enables genotype, ancestry, phenotype, and segment information to be combined within a single Python workflow without ad hoc table reconciliation.
Analysis modules			
Population structure and visualization	Genotype matrices with optional locus-specific ancestry masks and labels.	Standard PCA, missing-data PCA, multi-array ancestry-specific MDS, ancestry bar plots, chromosome painting, cohort heatmaps, Manhattan/Q–Q plots, and scatter plots.	Supports exploratory and publication-ready visualization of population structure from the same objects used for downstream modeling.
Association and population genetics	Genotypes, locus-specific ancestry calls, phenotypes, covariates, population labels, and allele-frequency summaries.	Variant-level GWAS, admixture mapping, D, f_2, f_3, f_4, F_{ST}, Tsallis F_q, f_4-ratio, block-jackknife errors, pseudo-haploid handling, and optional ancestry masking.	Connects association testing and allele-frequency statistics to the same ancestry-aware data model, enabling stratified analyses without splitting datasets by hand.
Simulation, IBD, and relatedness	Phased founder haplotypes, sample metadata, genetic maps, locus-specific ancestry, and IBD segments.	Haplotype-based admixed mosaic simulation, metadata propagation, IBD filtering, and ancestry-restricted segment trimming.	Provides test data, ancestry-aware relatedness summaries, and species-agnostic simulation workflows for diploid datasets.
Performance, reproducibility, and deployment			
Interfaces, performance, and reproducibility	The same object model exposed through both Python and a command-line interface.	Vectorized NumPy routines, memory-aware loading, streaming readers, optional PyTorch/GPU acceleration, compressed outputs, notebooks, CI-tested releases, and versioned documentation.	Supports reproducible interactive analyses, scripted pipelines, and high-performance computing workflows while reducing memory pressure on biobank-scale datasets.

Architecture

Core data model

The `snputils` architecture employs a modular design based on typed containers that keep numerical arrays aligned with the sample, variant, ancestry, phenotype, or segment metadata needed to interpret them. The `SNPObject` class serves as the primary container for genotype data, while `GRGObject` stores genotype representation graph data and can materialize a dense `SNPObject` when required. Ancestry-related information is managed through two specialized classes: the `LocalAncestryObject`, which stores local ancestry, more descriptively known as locus-specific ancestry, as per-window ancestry assignments for individual haplotypes, and the `GlobalAncestryObject`, which stores global ancestry, also known as genomic ancestry proportions, as per-individual admixture proportions (Q) and optional per-variant ancestry allele frequencies (P). A central design principle

Table 2. Feature comparison across genomic and ancestry-analysis tools. Check marks indicate native or general-purpose support for the feature, whereas crosses indicate no native or broadly applicable support.

Feature	snputils	PLINK	EIGENSOFT	GCTA	ADMIXTOOLS	BCFtools	VCFtools	Hail	scikit-allel	pgenlib	pysnptools	sgkit	cyvcf2
Native Python interface	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓	✓
Multi-format support	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
Genomic ancestry-proportion data support	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Locus-specific ancestry data support	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Identity-by-descent data support	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✗
Ancestry-specific PCA	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Single-ancestry SNP simulation	✓	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✓	✗
Multi-ancestry SNP simulation	✓	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗
Built-in statistical analyses	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗
Admixture mapping	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Advanced visualization tools	✓	✗	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗
Scales to biobank data	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓	✗
Data querying support	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓

is that numerical arrays remain explicitly linked to sample identifiers, variant coordinates, allele labels, ancestry-window coordinates, and phenotype identifiers, so that filtering, merging, and downstream analyses preserve metadata alignment throughout the workflow. Additionally, `PhenotypeObject` and `MultiPhenotypeObject` support continuous and categorical trait data, while `IBDObject` stores identity-by-descent information for relationship and demographic inference. These containers expose Pythonic access patterns and array-based interoperability with NumPy (Harris et al., 2020), SciPy (Virtanen et al., 2020), and scikit-learn (Pedregosa et al., 2011), with optional PyTorch-backed computation for selected matrix operations (Paszke et al., 2019). Core genotype and ancestry containers also support object-level persistence for efficient reuse across analysis stages.

Multi-format I/O and data conversion

The library supports widely used file formats commonly encountered in population-genetic and genetic-association workflows through specialized readers that load data into unified internal structures. All delimited and whitespace-based text formats accept plain, gzip-compressed (`.gz`), or Zstandard-compressed (`.zst`) input, while binary formats are read directly in their native representations. Genotype data from VCF files (Danecek et al., 2011), BCF, PLINK 1 `bed/bim/fam` filesets (Purcell et al., 2007), PLINK 2 `pgen/pvar/psam` filesets (Chang et al., 2015), and BGEN files (Band and Marchini, 2018; Bycroft et al., 2018) are parsed into `SNPObject` containers. VCF and BCF readers also support annotation-only files with no sample or genotype columns, preserving variant-level metadata and INFO annotations so resources such as gnomAD-like frequency VCFs can be loaded directly. Genotype Representation Graph (GRG) files (DeHaas et al., 2025) are parsed into `GRGObject` containers. Locus-specific ancestry calls in `msp` format, including outputs compatible with RFMix v2 (Maples et al., 2013) and Gnomix (Hilmarsson et al., 2026), FLARE locus-specific ancestry VCF output (Browning et al., 2023), and `.lanc` file format (Hou et al., 2024) are parsed into `LocalAncestryObject` containers. Genomic ancestry-proportion matrices in Q/P format produced by ADMIXTURE (Alexander et al., 2009; Alexander and Lange, 2011), Neural ADMIXTURE (Dominguez Mantes et al., 2023), or ADAMIXTURE (Saurina-i Ricos et al., 2026) are parsed into `GlobalAncestryObject` containers. Additionally, the library provides corresponding writers and conversion utilities for all supported formats, enabling data export and format conversion without leaving Python. The reader system employs automatic file format detection and provides standardized access patterns that are independent of the underlying file structure. This design eliminates the need for users to write format-specific parsing implementations and reduces potential sources of error in analytical pipelines. The readers are optimized for high-throughput data extraction, enabling selective retrieval of genomic regions, sample subsets, variant subsets, or specific fields without requiring full dataset materialization in memory. Readers expose chunked iteration interfaces for streaming analyses, while optimized tabular parsing accelerates common genotype-only and metadata-selection workloads. Memory-efficient data structures and chunk-aware algorithms further reduce RAM usage during file operations, allowing processing of large-scale datasets that would otherwise exceed system memory limits. The writer system extends these optimizations to output operations, including VCF chromosome partitioning and standard BCF, BED, PGEN, BGEN, `msp`, FLARE VCF, `lanc`, and Q/P export. Because chromosome and contig labels are retained in the objects, the same I/O layer natively handles non-human genomes. Non-autosomal regions, including sex chromosomes and mitochondrial DNA, are stored using the same genotype attributes as autosomes. For haploid calls represented in diploid-formatted files, one allele may be observed while the other is missing; when converted to dosage, reference/missing calls are encoded as 0, alternate/missing calls as 2, and fully missing calls remain missing. For human datasets, chromosome-label normalization is implemented as an optional convenience utility rather than a requirement for downstream analysis.

For identity-by-descent data, `snputils` auto-detects hap-IBD (Zhou et al., 2020) and ancIBD (Ringbauer et al., 2024) outputs and parses them into `IBDObject` containers. All IBD parsers use optimized tabular parsing for efficient loading and standardize fields into a common schema.

Manipulation and quality-control routines are provided across the genetic objects, including variant and sample subsetting with sample-metadata alignment; duplicate sample, variant-ID, and coordinate checks; call-rate, minor allele frequency (MAF), minor allele count (MAC), Hardy-Weinberg equilibrium (HWE), differential-missingness, imputation-quality, and LD-pruning filters; heterozygosity and inbreeding summaries and outlier reports; reported-vs-genetic sex checks; chromosome-name normalization; dataset intersection, merge, and concatenation operations; removal of strand-ambiguous variants; and detection and correction of flipped or mismatching alleles across datasets.

Listing 1. Quickstart example of a snputils workflow.

```python
import snputils as su

snp = su.read_snp("cohort.vcf.gz")      # Read VCF/BCF, PLINK, BGEN,...
snp = snp.filter_biallelic_variants()   # filter to biallelic
snp.save("cohort.pgen")                 # Save as pgen or other format

af = snp.allele_freq()                  # allele frequencies (AF)
pcs = su.PCA(snp)                       # PCA

lai = su.read_lai("local_ancestry.msp") # LAI (msp, FLARE, or lanc)
adm = su.read_admixture("admixture")    # global ancestry
labels = su.read_labels("labels.tsv")   # sample metadata

afr_af = snp.allele_freq(ancestry="AFR", laiobj=lai) # AFR-specific
su.viz.chromosome_painting(lai, "chr_paintings/")    # chr painting

mdpca = su.mdPCA(snp, lai, labels, ancestry="AFR")   # mdPCA
su.viz.scatter(mdpca, labels)                        # plot mdPCA

pheno = su.read_pheno("phenotypes.tsv", col="trait") # read pheno
ibd = su.read_ibd("hap.ibd")                         # read IBD

gwas = su.run_gwas(pheno, snp)                # GWAS
admix = su.run_admixture_mapping(pheno, lai) # admixture mapping
su.viz.qq_plot(gwas)                         # Q-Q plots
su.viz.qq_plot(admix)
su.viz.manhattan_plot(gwas)                  # Manhattan plots
su.viz.manhattan_plot(admix)
```

These harmonization utilities are particularly important in multi-cohort studies, where small inconsistencies in genome build conventions, chromosome labels, allele orientation, and missingness filters can otherwise propagate into downstream analyses.

In addition to the Python API, `snputils` installs a `snputils` command-line interface (CLI) for file-oriented workflows that can run directly in shell scripts, high-performance computing jobs, or workflow managers. The CLI exposes subcommands for standard PCA, missing-data PCA, multi-array ancestry-specific MDS, admixture mapping, GWAS, haplotype-based admixture simulation, and visualizations. Examples are shown in Listing 2.

File-format reading performance

We compared `snputils` readers for BED, PGEN, VCF, BCF, and BGEN against pgenlib (Chang et al., 2015), bgen (McRae, 2026), pysnptools (Kadie and Heckerman, 2017), pandas-plink (Horta et al., 2025), sgkit (sgkit developers, 2021), plinkio (Frånberg et al., 2024), cyvcf2 (Pedersen and Quinlan, 2017), scikit-allel (Miles et al., 2020), pysam (Heger et al., 2026), Hail (Poterba et al., 2025), and PyVCF3 (dridk, 2026) using Chromosome 22 from the 1000 Genomes Project (The 1000 Genomes Project Consortium, 2015).

For each format, we measured wall-clock time and peak resident memory for ingesting the file and materializing the genotype array within a Python environment for interactive analysis. We evaluated summed alternate-allele dosages and, where supported, explicit phased diploid allele calls. and genotype probability tensors for the BGEN format. For each completed run, we verified that all readers returned the same genotype array; this verification step was performed outside the timed and memory-profiled region. As shown in Figure 1, `snputils` was the fastest or tied for fastest and among the most memory-efficient readers for BED and PGEN, while it was the fastest and the most memory-efficient reader for VCF, BCF, and BGEN. For summed BED/PGEN dosage reads, it was effectively tied with direct pgenlib calls in runtime and used similar peak memory. The explicit allele-call advantage comes from grouping variants into contiguous chunks and limiting the large low-level allele buffers to those chunks before materializing the compact output representation, reducing both conversion overhead and peak memory relative to basic pgenlib usage. For VCF and BCF, the `snputils` reader avoids unnecessary tabular materialization, yielding substantially lower runtime and memory than scikit-allel, cyvcf2, pysam, and other VCF readers. Methods whose read time exceeded 60 minutes were stopped and were not included in the peak-memory panels, because peak-memory values are only comparable for completed full reads.

GPU acceleration

Beyond CPU-based optimizations, `snputils` incorporates optional GPU acceleration through PyTorch integration to improve computational performance on large genomic datasets. The TorchPCA implementation leverages CUDA-enabled computation for principal component analysis and supports both exact and approximate low-rank decompositions. This functionality is particularly useful when population-structure analyses must be repeated across large genotype matrices, ancestry masks, or multiple parameter settings. The concordance and runtime benchmarks below evaluate these accelerated implementations against established reference tools.

Listing 2. Examples of snputils CLI usage.

```
# PCA from a VCF, BCF, BGEN, bed, or pgen fileset
snputils pca \
  --snp-path cohort.pgen \
  --plot pca.png \
  --coords pca.tsv

# Ancestry-aware embeddings can be launched from workflow scripts
snputils mdpca \
  --snp-path cohort.pgen --lai-path local_ancestry.msp \
  --labels-file labels.tsv --ancestry AFR \
  --coords mdpca.tsv --plot mdpca.png

# Association workflows from pheno + genotype or LAI files
snputils gwas \
  --phe-id trait --phe-path phenotypes.tsv \
  --snp-path cohort.pgen --results-path gwas.tsv.gz \
  --manhattan-plot gwas_manhattan.png --qq-plot gwas_qq.pdf

snputils admixture-map \
  --phe-id trait --phe-path phenotypes.tsv \
  --lai-path local_ancestry.msp --results-path admixmap.tsv.gz \
  --manhattan-plot admixmap_manhattan.svg --qq-plot admixmap_qq.svg

# Simulation
snputils simulate \
  --snp founders.vcf.gz --metadata founders.tsv \
  --output-dir simulated_batches/ --n-batches 10
```

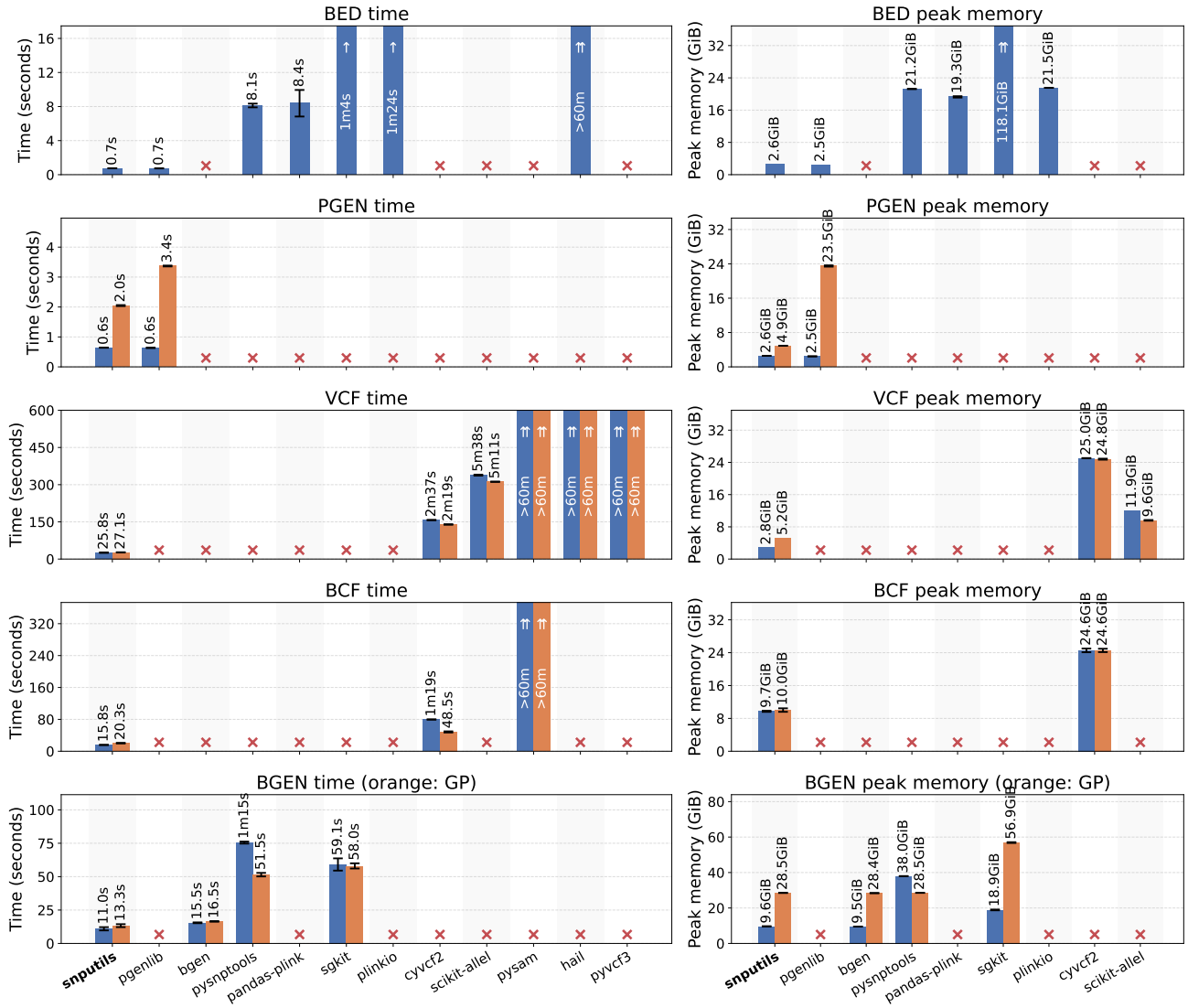

Figure 1 Time and peak memory usage comparison across formats and tools for reading Chromosome 22 SNPs for all samples in the 1000 Genomes Project (The 1000 Genomes Project Consortium, 2015). Blue bars show summed alternate-allele dosages; orange bars show phased allele codes except for BGEN, where they show genotype probability tensors. For each condition, error bars show standard deviations across three repeated runs; wall-clock time and peak memory were measured in separate runs because memory profiling changes runtime, and repeated reads in the same Python process can retain allocator state that distorts resident-memory estimates.

Alt text: A ten-panel benchmark figure compares read time and peak memory for BED, PGEN, VCF, BCF, and BGEN readers. Each format has a time panel and a memory panel, with paired bars where both genotype representations are supported. Blue bars show summed alternate-allele dosages; orange bars show phased allele codes except for BGEN, where they show genotype-probability tensors. `snputils` is fastest and uses the least memory overall, and appears tied with direct pgenlib for summed BED/PGEN dosage timing. Several slow VCF methods are marked as exceeding 60 minutes in the time panel and are omitted from the memory panel because they did not complete a comparable full-read memory profile.

Population-genetic analyses and ancestry-aware workflows

Admixed haplotype simulation

`snputils` includes a lightweight haplotype-based simulator (Geleta et al., 2026a) that generates mosaics of haplotypes from real phased founder haplotypes. Given a phased genotype file or `SNPObject`, per-sample metadata, and optionally a genetic map, the simulator samples a generation-since-admixture parameter for each batch of simulated individuals. Recombination breakpoints are then placed along the supplied SNP sequence, with probabilities proportional to the local genetic distance if a map is supplied, or uniformly across SNP indices otherwise. At each breakpoint, it switches the donor haplotype by permuting suffix segments across the selected batch of founder haplotypes, which yields tract length distributions that scale with genetic length and the generation

parameter when a genetic map is supplied. Both discrete labels (e.g., population codes) and continuous labels (e.g., latitude and longitude, optionally stored as 3D unit vectors) are propagated through the mosaic and can be aggregated per SNP window by an exact mode for discrete labels, or by the mean or an approximate GPU-friendly mode for continuous labels. This "select and stitch" approach is consistent with the standard Poisson model of crossovers used in prior simulation work (Karavani et al., 2019), while remaining simple to run at scale through a CLI that batches simulations and writes compressed NumPy archives, `SNPObject` pickles, or phased genotype outputs in the supported formats. The simulator is species-independent for diploid datasets: it operates on phased founder haplotypes and, where needed, a suitable genetic map or uniformly across SNP positions, rather than on human-specific annotations.

Population-genetic statistics

`snputils` provides a unified suite of population-genetic statistics, including D (Durand et al., 2011), f_2, f_3, f_4, the f_4-ratio (Patterson et al., 2012), the fixation index (F_{ST}) with support for Hudson (Hudson et al., 1992), Weir-Cockerham (Weir and Cockerham, 1984), and Tsallis F_q (Geleta et al., 2026c) methods. Each estimator is computed from per-SNP contributions aggregated over blocks, and accompanied by delete-one block jackknife standard errors (Reich et al., 2009). Inputs can be either aggregated allele frequency and count matrices with population names or a `SNPObject` plus per-sample labels, which are internally converted to population-level allele frequencies. All allele-frequency-based statistics support ancestry-specific analyses (see below) by masking genotypes to a selected locus-specific ancestry using a `LocalAncestryObject` before aggregation, which enables ancestry-stratified tests without pre-splitting datasets. The same allele-frequency backend can export blocked pairwise statistics in formats compatible with qpAdm, qpWave, and qpGraph, allowing `snputils` preprocessing and ancestry-masked outputs to feed downstream ADMIXTOOLS-style modeling. The allele-frequency aggregation layer can also detect and handle pseudo-haploid samples, supporting ancient-DNA and other low-coverage settings that are common in molecular evolutionary studies. The routines are vectorized, tolerant to missingness, and scale to biobank data.

Ancestry and association analyses

The library supports population-genetic methods and data types related to genetic ancestry analysis. These include parsers for both genomic ancestry-proportion and locus-specific ancestry inference outputs, alongside downstream tools such as missing-data PCA (mdPCA) (Agrawal et al., 2026), multi-array ancestry-specific MDS (Aounallah et al., 2026), ancestry-specific computation of allele statistics, locus-specific ancestry visualization, variant-level GWAS, and admixture mapping. `snputils` also supports bidirectional conversion between SNP-level and window-level locus-specific ancestry representations, including aggregation into fixed-size SNP windows or explicit genomic intervals and projection of window-level ancestry back onto SNP coordinates.

Genomic ancestry proportions from Q and P format files are supported. The library supports merging multiple runs and visualizing results as ordered bar plots, with samples grouped by metadata and displayed using configurable color palettes to facilitate interpretation of population composition. For locus-specific ancestry, chromosome-painting functions render ancestry tracts along chromosomes for individual samples and generate cohort-level heatmaps summarizing ancestry frequencies across genomic windows. These visualizations support detailed examination of fine-scale admixture patterns and tract distributions across populations. These capabilities integrate with the f-statistics module, which can optionally restrict computations to a specified locus-specific ancestry to test hypotheses within ancestry components.

A shared association engine supports both variant-level GWAS and admixture mapping. For GWAS, `snputils` streams SNP dosages and fits additive linear models for quantitative traits and logistic regression models for binary traits, with optional covariate adjustment. For admixture mapping, the same engine performs locus-by-locus regression of locus-specific ancestry dosage against quantitative or binary traits. Both modes support phenotype and sample alignment, keep/remove sample lists, covariate adjustment with optional variance standardization, confidence intervals, Bonferroni and Benjamini–Hochberg corrections, compressed tabular outputs, and configurable memory limits that determine effective chunk sizes during computation. The admixture mapping pipeline accepts locus-specific ancestry calls and phenotype tables as input and produces PLINK-style association outputs while operating directly on the same locus-specific ancestry representation used by downstream ancestry-specific analyses.

Identity-by-descent and relatedness

`snputils` includes a unified identity-by-descent module that reads hap-IBD (Zhou et al., 2020) and ancIBD (Ringbauer et al., 2024), including outputs designed for ancient-DNA analyses, into a common `IBDObject` representation with automatic format detection. Convenience methods support fast filtering by chromosome, individuals, segment type, and minimum length. `SNPObject` also supports a variety of relatedness methods, including GRM-based relatedness, IBD-derived relatedness summaries, related-pair flagging, and deterministic related-sample pruning. For ancestry-aware analyses in admixed cohorts, ancestry masking can be performed by intersecting with a `LocalAncestryObject` to isolate IBD segments within intervals where both individuals carry a specified locus-specific ancestry. The restriction is haplotype-aware when haplotype IDs are present and can optionally require both haplotypes to match the target ancestry. Segments are either entirely filtered out or clipped into ancestry-consistent subsegments aligned to ancestry window boundaries, with cM lengths recomputed from genetic positions when available. These capabilities enable ancestry-specific relatedness screening and demographic inference and integrate naturally with downstream admixture and association workflows.

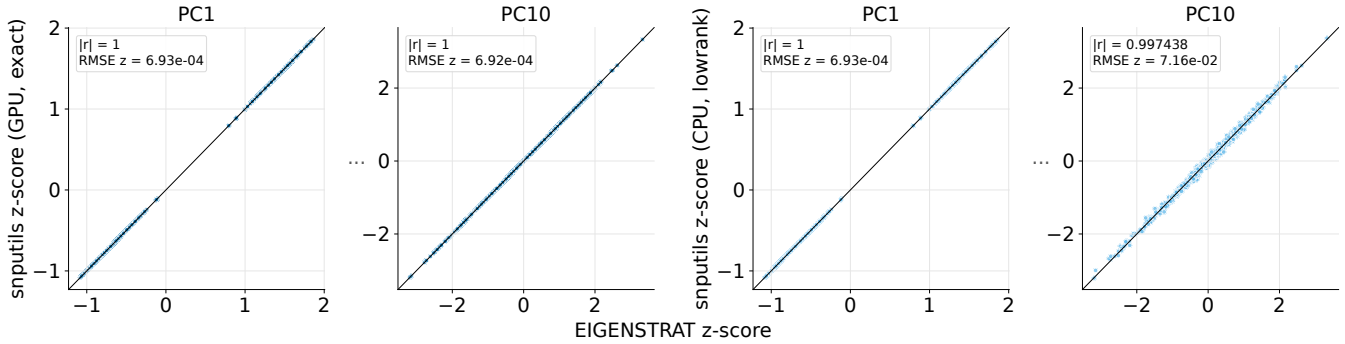

Figure 2 Concordance of standardized PCA sample coordinates from `snputils` and EIGENSOFT `smartpca`. The left panels compare `snputils` GPU exact PCA with EIGENSOFT `smartpca`, and the right panels compare `snputils` CPU low-rank PCA with EIGENSOFT `smartpca`. For each method, component scores were standardized within each principal component and aligned by sign before comparison. Concordance is shown for the first and tenth principal components using matched sample coordinates, with agreement summarized by the absolute Pearson correlation, root mean squared difference in standardized scores, and maximum absolute standardized difference.

Alt text: Grids of four scatter plots compare standardized principal component coordinates from `snputils` against EIGENSOFT `smartpca`. Both the GPU exact and CPU low-rank `snputils` implementations align closely with the identity line for most components, showing near-perfect agreement with the reference method.

Engineering, documentation, and extensibility

The `snputils` library adheres to modern software engineering practices, with comprehensive dependency management and testing frameworks. The library can be installed with `pip install snputils` and supports maintained Python releases, with continuous integration that runs the test suite across the oldest and newest supported Python versions and builds the documentation. Unit and regression tests cover readers, transformations, statistics, command-line tools, and plotting-oriented workflows. Documentation is versioned alongside releases and includes tutorials and example notebooks designed to facilitate user adoption and reproducible analysis. Public issue tracking and feature-request workflows further support community feedback, bug reporting, and iterative development. The open-source nature of the project, distributed under a BSD 3-Clause License, promotes community adoption and contribution.

Visualization and plotting utilities

`snputils` includes a visualization layer for exploratory analysis, quality control, and publication-oriented summaries. Supported plots include principal-component scatter plots, admixture-proportion bar plots, chromosome paintings at both individual and cohort levels, Manhattan plots for GWAS and admixture mapping results, and other summaries. The visualization layer is designed for both interactive notebook workflows and scripted analyses, with support for customizable color palettes, annotation layers, sample ordering, and output paths. Because the visualization utilities operate directly on the same objects used throughout computation and analysis, users can generate exploratory and manuscript-ready figures without exporting intermediate data into separate visualization scripts.

Concordance and performance benchmarks

To evaluate end-to-end analytical performance beyond file ingestion, we benchmarked `snputils` across three analysis classes commonly used in population-genetic and genetic-association studies: PCA, allele-frequency-based population statistics, and admixture mapping. The benchmarks were designed to assess both numerical concordance with widely used reference tools and wall-clock runtime on identical input datasets. Concordance was evaluated using matched outputs from each method, while performance was measured as elapsed runtime for the corresponding analysis workflow. All runtime benchmarks were repeated five times on a compute node equipped with 16 CPUs (AMD EPYC 7713 64-Core Processor). CPU-based PCA and allele-frequency statistics benchmarks used 20 GB of memory, whereas GPU-accelerated PCA benchmarks were performed using an NVIDIA A100 GPU.

Principal components analysis

PCA benchmarks were performed using the Phase 3 1000 Genomes Project panel (The 1000 Genomes Project Consortium, 2015). We selected 50 individuals from each of twelve populations (YRI, LWK, MSL, GWD, CEU, FIN, GBR, IBS, CHB, JPT, GIH, and PEL), yielding 600 individuals, and retained 100,000 SNPs. The benchmark compared the leading twelve principal component coordinates from `snputils` with those generated by EIGENSOFT `smartpca` (Patterson et al., 2006). Two PCA implementations in `snputils` were evaluated: a CPU-based low-rank decomposition intended for large genotype matrices where approximate factorization is desirable, and an exact GPU-based decomposition.

Because principal components are identifiable only up to sign and scale, scores for each principal component were independently standardized across samples and aligned by sign prior to comparison. Concordance for the first twelve principal components was summarized using the absolute Pearson correlation, root mean squared difference in standardized

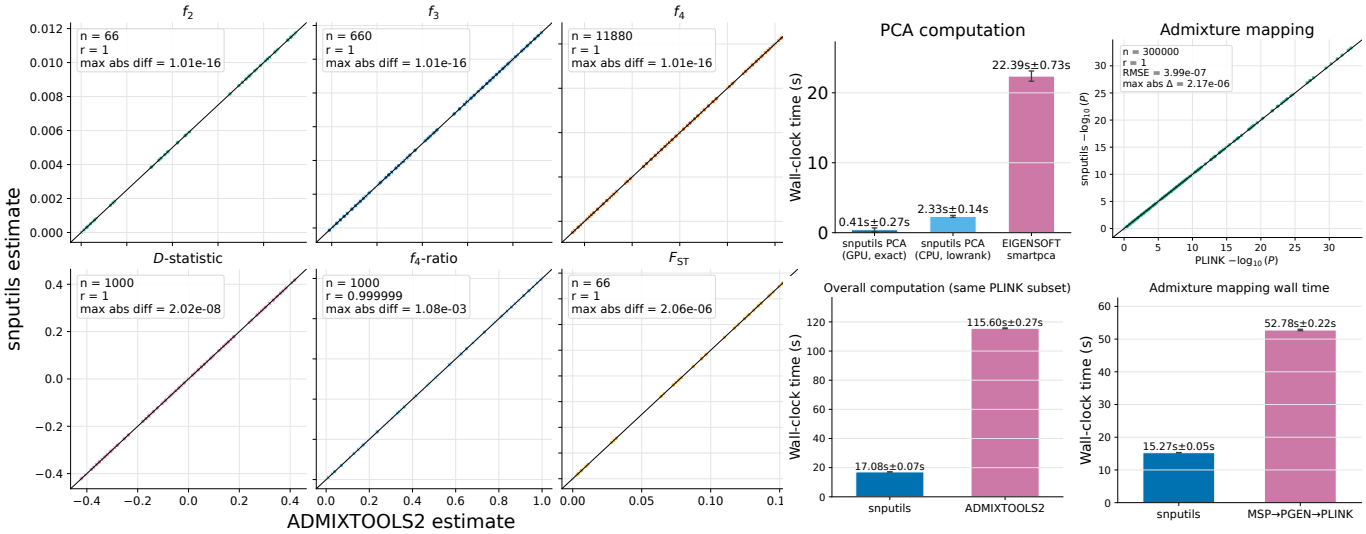

Figure 3 Concordance and runtime benchmarks for allele-frequency statistics, PCA, and admixture mapping. The large left panel shows concordance of allele-frequency-based population statistics computed by `snputils` and ADMIXTOOLS 2, including f_2, f_3, f_4, D statistic, f_4-ratio, and F_{ST} estimates. Points represent matched population configurations, with agreement assessed against the identity line. In the right block, the upper-left panel shows wall-clock runtime for PCA computation using EIGENSOFT `smartpca`, `snputils` CPU low-rank PCA, and `snputils` GPU exact PCA. The lower-left panel shows wall-clock runtime for the complete allele-frequency statistics benchmark using the same PLINK subset on 16 CPUs with 20 GB memory. The upper-right panel shows concordance of admixture mapping association evidence from `snputils` and PLINK 2, comparing matched $-\log_{10}(P)$ values across ancestry-window tests. The lower-right panel shows wall-clock runtime for admixture mapping on 16 CPUs with 60 GB memory. Runtime bars show mean ± standard deviation across five repetitions.

Alt text: A multi-panel benchmark figure combines six concordance scatter plots for allele-frequency statistics with four smaller plots for PCA runtime, allele-frequency statistic runtime, admixture mapping concordance, and admixture mapping runtime. The panels show near identity-line agreement with reference tools and shorter wall-clock times for `snputils`.

scores, and maximum absolute standardized difference. The PCA concordance results shown in Figure 2 demonstrate that `snputils` closely reproduces EIGENSOFT coordinates across the leading ancestry axes, with absolute correlations approaching 1 for both CPU-based low-rank PCA and GPU-based exact PCA. The corresponding runtime benchmarks are shown in Figure 3. Relative to EIGENSOFT, `snputils` CPU low-rank PCA achieved a 9.6-fold reduction in runtime, while GPU exact PCA achieved a 54.6-fold reduction.

Allele-frequency statistics

We next evaluated allele-frequency-based population statistics against ADMIXTOOLS 2 (Patterson et al., 2012; Maier et al., 2023) using the same 1000 Genomes phase 3 population panel (The 1000 Genomes Project Consortium, 2015). The benchmark included 50 individuals per population and retained up to 1,000,000 SNPs. Comparisons were performed for pairwise f_2 statistics, outgroup and admixture f_3 statistics, ordered f_4 statistics, the D statistic (Durand et al., 2011), f_4-ratio estimates (Patterson et al., 2012), and Hudson's F_{ST} (Hudson et al., 1992). For both `snputils` and ADMIXTOOLS 2, standard errors and test statistics were estimated using non-overlapping 50 kb genomic blocks.

For f_2 and F_{ST}, all unordered population pairs were evaluated. For f_3, all target populations were tested against all unordered pairs of distinct reference populations. For f_4, all ordered population quadruples were considered. To keep the most computationally intensive analyses tractable while still sampling a broad range of population configurations, the benchmark used 1,000 ordered quadruples for D statistics and 1,000 five-population configurations for f_4-ratio estimation. For each statistic, estimates from `snputils` and ADMIXTOOLS 2 were matched by population configuration and compared using Pearson correlation, mean absolute difference, maximum absolute difference, and root mean squared difference. Across all six statistic classes, `snputils` produced estimates that were highly concordant with ADMIXTOOLS 2 (Figure 3). `snputils` completed the analysis 6.8-fold faster than ADMIXTOOLS 2.

Admixture mapping

Finally, we compared admixture mapping association results from `snputils` with PLINK 2 (Chang et al., 2015) using a synthetic locus-specific ancestry benchmark. Because PLINK does not operate directly on locus-specific ancestry interval calls, each ancestry-window dosage was encoded as a biallelic dosage-like predictor in PGEN format, allowing PLINK 2 to fit the same linear model to the same predictor matrix used by `snputils`. The benchmark used 10,000 samples, 100,000 locus-specific ancestry windows distributed across the autosomes, and three simulated ancestry states. Three quantitative covariates were included: two standardized genome-wide ancestry proportions, with one of the three omitted as the reference component, and an independent standardized Gaussian covariate. A quantitative phenotype was simulated from locus-specific ancestry at a subset of windows together with covariate effects and residual noise.

Both methods fit the same additive linear association model for each ancestry-window test while adjusting for the same covariates. Concordance was evaluated by matching tests

according to chromosome, genomic position, and ancestry assignment, and comparing $-\log_{10}(p)$ values. This benchmark therefore tests whether `snputils` recovers the same admixture-mapping association evidence as PLINK while operating on the locus-specific ancestry representation used throughout downstream ancestry-specific analyses. As shown in Figure 3, `snputils` and PLINK produced highly concordant admixture mapping p values, while `snputils` completed the analysis 3.5-fold faster than the ancestry-PGEN PLINK 2 workflow.

Discussion

In summary, `snputils` unifies genotype, ancestry, phenotype, IBD, population-statistic, simulation, visualization, GWAS, and admixture mapping workflows in a single Python library that scales to large cohort data. It combines efficient I/O and GPU-ready computation with ancestry-aware analytical methods, including missing-data PCA, multi-array ancestry-specific MDS, locus-specific ancestry masking for f-statistics, and ancestry-restricted IBD segment analysis. This breadth is important for molecular evolution and population genomics because many modern studies require the same cohort to move repeatedly between file conversion, quality control, ancestry inference, allele-frequency statistics, relatedness screening, and association testing. Although the benchmarks presented here use human data, the core data model and major analysis routines are species-agnostic for diploid datasets represented in supported standard formats. In practice, reusable components such as genotype I/O, PCA, allele-frequency statistics, GWAS, and simulation operate natively on non-human datasets, while human-oriented components, such as default chromosome-label normalization and locus-specific ancestry chromosome-painting visualizations, remain optional and modular. The `snputils` API reduces intermediary code and improves reproducibility for interactive and pipeline use by keeping genotype, ancestry, phenotype, IBD, simulation, visualization, GWAS, admixture mapping, and population-statistic workflows within a shared ecosystem. The benchmarks show that this integration does not come at the cost of performance: `snputils` achieves orders-of-magnitude faster genotype ingestion workloads and provides concordant analytical results with substantial speedups over established reference tools for PCA, allele-frequency statistics, and admixture mapping.

The library's efficient and accessible interfaces reduce practical barriers to large-scale genomic analysis and facilitate reproducible, collaborative research. Ongoing development will prioritize community-requested file formats and analytical modules while maintaining backward-compatible, versioned releases. The software, documentation, and example workflows are freely available at snputils.org. We encourage the community to adopt, test, and contribute to `snputils` for future studies in genetics, comparative population genomics, clinical analyses, and large-scale cohort investigations.

Data availability

Documentation, tutorials, and the API reference for `snputils` are available at docs.snputils.org. The software is available on GitHub and on PyPI under the BSD 3-Clause License. Benchmark scripts are available in the benchmark directory of the public repository. The 1000 Genomes Project data are openly available through the International Genome Sample Resource.

Acknowledgements

We thank the reviewers for their useful feedback. We also thank the `snputils` community, including early users, for code contributions, testing, bug reports, feature requests, documentation feedback, suggestions, and validation on diverse datasets. We are especially grateful to colleagues working on open-source implementations for emerging file formats. We also thank the developers and maintainers of the open-source scientific software ecosystems on which snputils builds, as well as the public data resources used for testing and benchmarking.

References

Abdellaoui A, Yengo L, Verweij KJ, Visscher PM. 2023. 15 years of GWAS discovery: realizing the promise. *The American Journal of Human Genetics*. 110:179–194.

Agrawal D, Barrabés M, Bonet D, Pourshafeie A, Rodríguez Rodríguez JE, Ioannidis AG. 2026. mdPCA: Dimensionality Reduction for Highly Missing Data in Population Genetics. *Submitted*.

Alexander DH, Lange K. 2011. Enhancements to the ADMIXTURE algorithm for individual ancestry estimation. *BMC Bioinformatics*. 12:1–6.

Alexander DH, Novembre J, Lange K. 2009. Fast model-based estimation of ancestry in unrelated individuals. *Genome Research*. 19:1655–1664.

All of Us Research Program Genomics Investigators . 2024. Genomic data in the All of Us Research Program. *Nature*. 627:340–346.

Aounallah K, Bonet D, Barrabés M, Agrawal D, Rodríguez Rodríguez JE, Ioannidis AG. 2026. maasMDS: Multi-array Ancestry-specific Multidimensional Scaling for Fine-scale Structure in Admixed Genomes. *In Press*.

Band G, Marchini J. 2018. BGEN: a binary file format for imputed genotype and haplotype data. *bioRxiv*. 308296.

Barberena-Jonas C, Medina-Muñoz SG, Cedillo-Castelán V, Sepúlveda-Morales T, Gonzaga-Jáuregui C, García-García L, Ioannidis AG, Moreno-Estrada A. 2026. Clinical genetic variation across Hispanic populations in the Mexican Biobank. *Nature Medicine*. 1–11.

Baumdicker F, Bisschop G, Goldstein D, Gower G, Ragsdale AP, Tsambos G, Zhu S, Eldon B, Ellerman EC, Galloway JG, et al. 2022. Efficient ancestry and mutation simulation with msprime 1.0. *Genetics*. 220:iyab229.

Browning SR, Browning BL. 2015. Accurate non-parametric estimation of recent effective population size from segments of identity by descent. *The American Journal of Human Genetics*. 97:404–418.

Browning SR, Waples RK, Browning BL. 2023. Fast, accurate local ancestry inference with FLARE. *The American Journal of Human Genetics*. 110:326–335.

Bycroft C, Freeman C, Petkova D, Band G, Elliott LT, Sharp K, Motyer A, Vukcevic D, Delaneau O, O'Connell J, et al. 2018. The UK Biobank resource with deep phenotyping and genomic data. *Nature*. 562:203–209.

Chang CC, Chow CC, Tellier LC, Vattikuti S, Purcell SM, Lee JJ. 2015. Second-generation PLINK: rising to the challenge of larger and richer datasets. *GigaScience*. 4:7.

Chatterjee N, Shi J, García-Closas M. 2016. Developing and evaluating polygenic risk prediction models for stratified disease prevention. *Nature Reviews Genetics*. 17:392–406.

Cook MB, Sanderson SC, Deanfield JE, Reddington F, Roddam A, Hunter DJ, Ali R. 2025. Our Future Health: a unique global resource for discovery and translational research. *Nature Medicine*. 31:728–730.

COVID-19 Host Genetics Initiative . 2021. Mapping the human genetic architecture of COVID-19. *Nature*. 600:472–477.

Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, Handsaker RE, Lunter G, Marth GT, Sherry ST, et al. 2011. The variant call format and VCFtools. *Bioinformatics*. 27:2156–2158.

Danecek P, Bonfield JK, Liddle J, Marshall J, Ohan V, Pollard MO, Whitwham A, Keane T, McCarthy SA, Davies RM, et al. 2021. Twelve years of SAMtools and BCFtools. *GigaScience*. 10:giab008.

DeHaas D, Pan Z, Wei X. 2025. Enabling efficient analysis of biobank-scale data with genotype representation graphs. *Nature Computational Science*. 5:112–124.

Dias JA, Chen T, Xing H, Wang X, Rodriguez AA, Madduri RK, Kraft P, Zhang H. Oct. 2025. Evaluating multi-ancestry genome-wide association methods: Statistical power, population structure, and practical implications. *The American Journal of Human Genetics*. 112:2493–2508.

Dominguez Mantes A, Mas Montserrat D, Bustamante CD, Giró-i Nieto X, Ioannidis AG. 2023. Neural ADMIXTURE for rapid genomic clustering. *Nature Computational Science*. 3:621–629.

dridk . PyVCF3: A Variant Call Format reader for Python, 2026. URL https://github.com/dridk/PyVCF3.

Durand EY, Patterson N, Reich D, Slatkin M. 2011. Testing for ancient admixture between closely related populations. *Molecular Biology and Evolution*. 28:2239–2252.

Feng YCA, Chen CY, Chen TT, Kuo PH, Hsu YH, Yang HI, Chen WJ, Su MW, Chu HW, Shen CY, et al. 2022. Taiwan Biobank: A rich biomedical research database of the Taiwanese population. *Cell Genomics*. 2.

Frånberg M, Kazuki T, Frei O, Jónsson H, McRae J, YeWenting . mfranberg/libplinkio, Nov. 2024. URL https://github.com/mfranberg/libplinkio.

Gaziano JM, Concato J, Brophy M, Fiore L, Pyarajan S, Breeling J, Whitbourne S, Deen J, Shannon C, Humphries D, et al. 2016. Million Veteran Program: A mega-biobank to study genetic influences on health and disease. *Journal of Clinical Epidemiology*. 70:214–223.

Geleta M, Mas Montserrat D, Ioannidis NM, Ioannidis AG. 2026a. Point cloud local ancestry inference (PCLAI): continuous coordinate-based ancestry along the genome. *bioRxiv*.

Geleta M, Montserrat DM, Giro-i Nieto X, Ioannidis AG. 2026b. Autoencoders for genomic variation analysis. *Genome Research*. 36:348.

Geleta M, Montserrat DM, Ioannidis AG. A Tsallis-Entropy Lens on Genetic Variation. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7447–7451, 2026c.

Haller BC, Ralph PL, Messer PW. 2026. SLiM 5: Eco-evolutionary simulations across multiple chromosomes and full genomes. *Molecular Biology and Evolution*. 43:msaf313.

Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, et al. Sept. 2020. Array programming with NumPy. *Nature*. 585:357–362.

Heger A, Marshall J, Jacobs K, Beauchamp K, Martin M, Wu D, Ryan D, Finkernagel F, van den Beek M, Pearson M, et al. pysam-developers/pysam, Feb. 2026. URL https://github.com/pysam-developers/pysam.

Hilmarsson H, Kumar AS, Barrabés M, Rastogi R, Bustamante CD, Mas Montserrat D, Ioannidis AG. 2026. Scalable high resolution ancestry deconvolution for genomic data. *Nature Communications*.

Horta D, Bolser D, B D, Aguet F. limix/pandas-plink, May 2025. URL https://github.com/limix/pandas-plink.

Hou K, Gogarten S, Kim J, Hua X, Dias JA, Sun Q, Wang Y, Tan T, Atkinson EG, others . 2024. Admix-kit: an integrated toolkit and pipeline for genetic analyses of admixed populations. *Bioinformatics*. 40:btae148.

Hudson RR, Slatkin M, Maddison WP. 1992. Estimation of levels of gene flow from DNA sequence data. *Genetics*. 132:583–589.

Ioannidis AG, Blanco-Portillo J, Sandoval K, Hagelberg E, Miquel-Poblete JF, Moreno-Mayar JV, Rodríguez-Rodríguez JE, Quinto-Cortés CD, Auckland K, Parks T, et al. 2020. Native American gene flow into Polynesia predating Easter Island settlement. *Nature*. 583:572–577.

Kadie C, Heckerman D. 2017. Ludicrous speed linear mixed models for genome-wide association studies. *bioRxiv*. 154682.

Karavani E, Zuk O, Zeevi D, Barzilai N, Stefanis NC, Hatzimanolis A, Smyrnis N, Avramopoulos D, Kruglyak L, Atzmon G, et al. 2019. Screening human embryos for polygenic traits has limited utility. *Cell*. 179:1424–1435.

Karczewski KJ, Solomonson M, Chao KR, Goodrich JK, Tiao G, Lu W, Riley-Gillis BM, Tsai EA, Kim HI, Zheng X, et al. 2022. Systematic single-variant and gene-based association testing of thousands of phenotypes in 394,841 UK Biobank exomes. *Cell Genomics*. 2.

Khera AV, Chaffin M, Aragam KG, Haas ME, Roselli C, Choi SH, Natarajan P, Lander ES, Lubitz SA, Ellinor PT, et al. 2018. Genome-wide polygenic scores for common diseases identify individuals with risk equivalent to monogenic mutations. *Nature Genetics*. 50:1219–1224.

Kurki MI, Karjalainen J, Palta P, Sipilä TP, Kristiansson K, Donner KM, Reeve MP, Laivuori H, Aavikko M, Kaunisto MA, et al. 2023. FinnGen provides genetic insights from a well-phenotyped isolated population. *Nature*. 613:508–518.

Maier R, Flegontov P, Flegontova O, Işıldak U, Changmai P, Reich D. 2023. On the limits of fitting complex models of population history to f-statistics. *eLife*. 12:e85492.

Maples BK, Gravel S, Kenny EE, Bustamante CD. 2013. RFMix: a discriminative modeling approach for rapid and robust local-ancestry inference. *The American Journal of Human Genetics*. 93:278–288.

McRae J. jeremymcrae/bgen, May 2026. URL https://github.com/jeremymcrae/bgen.

Miles A, Rodrigues MF, Ralph P, Harding N, Pisupati R, Rae S, Millar T. 2020. cggh/scikit-allel: v1.3.2. *Zenodo*. doi:

10.5281/zenodo.3976233.
Moreno-Grau S, Vernekar M, Lopez-Pineda A, Mas-Montserrat D, Barrabés M, Quinto-Cortés CD, Moatamed B, Lee MTM, Yu Z, Numakura K, et al. 2024. Polygenic risk score portability for common diseases across genetically diverse populations. *Human Genomics*. 18:93.
Nagai A, Hirata M, Kamatani Y, Muto K, Matsuda K, Kiyohara Y, Ninomiya T, Tamakoshi A, Yamagata Z, Mushiroda T, et al. 2017. Overview of the BioBank Japan Project: Study design and profile. *Journal of Epidemiology*. 27:S2–S8.
Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Advances in Neural Information Processing Systems, volume 32, pages 8024–8035, 2019.
Patterson N, Price AL, Reich D. 2006. Population structure and eigenanalysis. *PLoS Genetics*. 2:e190.
Patterson N, Moorjani P, Luo Y, Mallick S, Rohland N, Zhan Y, Genschoreck T, Webster T, Reich D. 2012. Ancient admixture in human history. *Genetics*. 192:1065–1093.
Pedersen BS, Quinlan AR. 2017. cyvcf2: fast, flexible variant analysis with Python. *Bioinformatics*. 33:1867–1869.
Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 12:2825–2830.
Perera M, Montserrat DM, Barrabés M, Geleta M, Giró-i Nieto X, Ioannidis AG. Generative moment matching networks for genotype simulation. In: 2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), pages 1379–1383. IEEE, 2022.
Phulka JS, Ashraf M, Bajwa BK, Pare G, Laksman Z. 2023. Current state and future of polygenic risk scores in cardiometabolic disease: a scoping review. *Circulation: Genomic and Precision Medicine*. 16:286–313.
Poterba T, Vittal C, King D, Goldstein D, Goldstein JI, Schultz P, Karczewski KJ, Seed C, Neale BM. 2025. The Scalable Variant Call Representation: Enabling Genetic Analysis Beyond One Million Genomes. *Bioinformatics*. 41.
Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MA, Bender D, Maller J, Sklar P, De Bakker PI, Daly MJ, et al. 2007. PLINK: a tool set for whole-genome association and population-based linkage analyses. *The American Journal of Human Genetics*. 81:559–575.
Reich D, Thangaraj K, Patterson N, Price AL, Singh L. 2009. Reconstructing Indian population history. *Nature*. 461:489–494.
Ringbauer H, Huang Y, Akbari A, Mallick S, Olalde I, Patterson N, Reich D. 2024. Accurate detection of identity-by-descent segments in human ancient DNA. *Nature Genetics*. 56:143–151.
Saurina-i Ricos J, Montserrat DM, Ioannidis AG. 2026. ADAMIXTURE: adaptive first-order optimization for biobank-scale genetic clustering. *Bioinformatics*. 42:btag236.
sgkit developers . sgkit: Statistical genetics toolkit in Python. https://sgkit-dev.github.io/sgkit/latest/, 2021. Software documentation.
Tanigawa Y, Li J, Justesen JM, Horn H, Aguirre M, DeBoever C, Chang C, Narasimhan B, Lage K, Hastie T, et al. 2019. Components of genetic associations across 2,138 phenotypes in the UK Biobank highlight adipocyte biology. *Nature Communications*. 10:4064.
The 1000 Genomes Project Consortium . Oct. 2015. A global reference for human genetic variation. *Nature*. 526:68–74.
Uffelmann E, Huang QQ, Munung NS, De Vries J, Okada Y, Martin AR, Martin HC, Lappalainen T, Posthuma D. 2021. Genome-wide association studies. *Nature Reviews Methods Primers*. 1:59.
Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, et al. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 17:261–272.
Weir BS, Cockerham CC. 1984. Estimating F-statistics for the analysis of population structure. *Evolution*. 1358–1370.
Yang J, Lee SH, Goddard ME, Visscher PM. 2011. GCTA: a tool for genome-wide complex trait analysis. *The American Journal of Human Genetics*. 88:76–82.
Zhou Y, Browning SR, Browning BL. 2020. A fast and simple method for detecting identity-by-descent segments in large-scale data. *The American Journal of Human Genetics*. 106:426–437.