IDEAL GENOM: Integrated Downstream Analytical Pipeline for Genomic Analysis

Luis Giraldo González Ricardo[1], Amabel M.M. Tenghe[1], Ashwin Ashok Kumar Sreelatha[1], and Manu Sharma[1]

[1]Centre for Genetic Epidemiology, Universitätsklinikum Tübingen, Tübingen, Germany

August 27, 2025

Abstract

Background: Since the release of the first human genome draft, there has been an exponential increase in the development of technologies that have offered unprecedented opportunities to unravel the genetic underpinnings of rare and complex diseases. However, despite the progress, the challenge persists. This includes the need for developing efficient computational tools to analyse large genomic datasets. The availability of clinical information, along with the development of cost-effective arrays that can scan hundreds of thousands of patients and healthy subjects to identify underlying risk variants, necessitates the development of tools that can streamline and automate the processing of millions of genetic markers.

Results: To address this challenge, we present IDEAL-GENOM (Integrated Downstream Analytical Pipeline for Genomic Analysis), a Python-based framework designed to streamline the analytical workflow commonly implemented in genome-wide association studies (GWAS) settings. IDEAL-GENOM integrates widely used tools such as PLINK and GCTA along with custom-developed functionalities, enabling reproducible results through parameter sharing. Initial testing demonstrates that our framework reduces quality control processing time by 10%, requiring minimal user intervention. Additionally, IDEAL-GENOM supports both local and cloud-based environments, enhancing its accessibility and scalability.

Conclusions: IDEAL-GENOM pipeline offers a simplified GWAS pipeline that will allow beginners and advanced users to leverage the in-built functionalities to perform GWAS analysis for complex diseases.

Keywords: GWAS, Python, summary statistics, visualization, quality control

1 Introduction

The advancement of genomic technologies has increased our ability to screen large cohorts, while parallel improvements in computational capabilities —including fast processors and expanded memory capacity— have made large-scale population-based genetic studies increasingly feasible [1]. This simultaneous development has enabled the scientific community to explore the underlying genetic underpinnings of both rare and complex diseases with unprecedented depth. However, despite the progress, the inadequate handling of low-quality data may lead to wrong conclusions, that necessitates the implementation of comprehensive quality control protocols to systematically identify and remove low-quality samples and single-nucleotide polymorphisms (SNPs) [2, 3].

Since the publication of the first-generation GWAS [4], it has become a mainstay in variant discovery, this involves the use of GWAS summary statistics that are required not only to confirm/validate the findings but also refining top loci, and additional downstream analysis, an aspect in which our toolkit also offers noticeable optimizations [5, 6, 7].

A multitude of computational tools have been developed to manage the complexities of large genomic datasets. However, integrating these tools into cohesive, reproducible workflows remains a persistent challenge. For instance, while many quality control steps can be performed using PLINK1.9, others require PLINK2.0 or a mix of scripts written in programming languages such as Perl or R. Similarly, GWAS can be performed using PLINK or GCTA, but downstream analyses often necessitate combining multiple tools in an unstructured manner [8, 9, 10, 11]. This fragmented approach complicates reproducibility because manually combining scripts can lead to error propagation.

To address these challenges, we introduce IDEAL-GENOM, a free and open-source Python toolkit designed to integrate and streamline genomic data analysis. IDEAL-GENOM offers a seamless workflow for performing quality control and GWAS analysis. The analytical workflow is enriched with multiple visualization functionalities to allow users to control how the pipeline behaves. Our toolkit minimises dependencies in the sense that it is completely written in Python, while offering customization, publication-ready plots and easy-to-use example notebooks to guide users through its functionalities. By emphasizing simplicity, flexibility, and reproducibility, IDEAL-GENOM empowers researchers to focus on generating insights rather than navigating technical hurdles.

In this paper, we describe the structure of IDEAL-GENOM and the philosophy that led to its design and development. We also demonstrate the functionalities of its constituent libraries while showcasing their advantages.

2 Implementation and design

2.1 Overview

2.1.1 Description

IDEAL-GENOM provides an easy-to-use workflow for genomic quality control (QC) and GWAS pipelines. Rather than rebuilding computational tools from scratch, IDEAL-GENOM leverages the strength of widely used and well-tested software; in other words, our toolkit provides a unified framework to perform a wide range of tasks related to the GWAS data analysis pipeline. The current release of the toolkit also includes two distinct libraries: The `ideal-genom-qc` library provides all the necessary functions for genomic QC, including the removal of low-quality samples, unreliable markers, and ancestry outliers, where each one of these steps is complemented with multiple high-quality plots that offer better insights into each stage. Furthermore, it also supports population structure exploration using the Uniform Manifold Approximation and Projection (UMAP) algorithm [12], which enables a more detailed overview of the genetic architecture than the one provided using only PCA, then paving the way for analysis of ethnically diverse populations. The second one is called `ideal-genom` and is a library dedicated to GWAS and generates two types of GWAS summary statistics: one based on a generalized linear model (GLM) and another using a generalized linear mixed model (GLMM). For GLM, `ideal-genom` is based on PLINK2, while for GLMM, it uses GCTA. Moreover, it also contains functionalities to generate high-quality plots to illustrate the results encoded in GWAS summary statistics.

The decision to develop flexible visualization functionalities was a key factor in choosing Python

as the foundation for our toolkit. This choice was driven not only by the availability of high-quality plotting libraries, such as `seaborn` and `matplotlib`, but also by the robust data manipulation capabilities of `pandas`, which are essential for interpreting and transforming the summary statistics files. IDEAL-GENOM offers complete flexibility and control to the users vis-à-vis the use and application of the two libraries. The two libraries are designed to function independently, allowing users to choose based on their specific requirements. By separating the software into distinct packages, users can avoid installing unnecessary components. This modular design also simplifies maintenance and updates, ensuring that future improvements of packages are independent of each other.

The toolbox is designed for seamless integration into broader analytical workflows. The command-line interfaces (CLI) facilitate the incorporation of both libraries into automated pipelines and bash scripts with minimal overhead. Its modular architecture allows researchers to employ the entire framework or selectively incorporate specific components into larger, more complex projects. Furthermore, a Dockerfile has been created to simplify the deployment on computing clusters. Pipeline parameters are managed via `JSON` configuration files rather than being passed directly through the CLI. This design choice improves reproducibility, especially for research teams working in multiple locations.

IDEAL-GENOM pipeline has been developed keeping in mind to train young researchers who are beginners and also it offers options for advanced users to control analytical downstream analyses based on their requirements. Each package is accompanied by a set of Jupyter notebooks designed to facilitate users' understanding of the toolkit's functionalities. These notebooks are particularly valuable for beginners, offering a hands-on approach that enables users to engage directly with the pipeline. While these notebooks are useful for newcomers, they also serve as a valuable resource for more advanced users, allowing them to explore the libraries' customization capabilities in depth.

2.1.2 `ideal-genom-qc`

`ideal-genom-qc` is a package whose main goal is to perform genomic QC using PLINK1.9 and PLINK2. In its current version, it processes raw data in PLINK binary format (bed, bim, fam); however, future updates aim to support a broader range of file types commonly used in genetic studies. Additionally, the package leverages the functionalities of well-known Python libraries such as `matplotlib` and `seaborn` to generate high-quality plots.

This library is structured to sequentially execute sample QC, ancestry checks, and variant QC. The ancestry check includes visually informative graphs that facilitate the basic analysis of population structure using PCA dimensionality reduction techniques. Currently, this feature is optimized for homogeneous populations, such as those represented in the 1000 Genomes Project (1000G) superpopulations [13]. This step is complemented with a more detailed study of the population structure relying on the UMAP dimensionality reduction technique.

The simultaneous visualization of quality control steps allows researchers to assess whether the chosen QC parameters are optimally suited for their analysis.

2.1.3 `ideal-genom`

This library is designed to perform a wide range of tasks, which can be categorized into three main functions:

1. Processing and preparing imputed genomic data,
2. Performing GWAS using GLM and GLMM, and

3. Generating various visualization tools for summary statistics interpretation.

For processing imputed genomic data, the library utilizes bcftools and PLINK2 [14]. GWAS analyses are performed using PLINK2 for GLM models, while GCTA is employed for GLMM models. The package supports several widely used visualization tools commonly employed by the genetics community to report findings. These visualizations are generated using the capabilities of `matplotlib` and `seaborn`. While similar visualization tools are available in R, we chose to develop own functions in Python to ensure compatibility and easier maintenance.

The `ideal-genom` pipeline seamlessly executes the necessary steps to process imputed data, including decompressing password-protected `zip` files, processing them until getting PLINK binary files; training GLM and GLMM models. Visualization tools are not directly integrated into the main pipeline because fine-tuning plots typically require multiple iterations to optimize parameters. To address this, we have provided example Jupyter notebooks that serve as a walk-through of the library's basic visualization capabilities.

3 Workflows

In this section, we describe the full workflow implemented in the IDEAL-GENOM software suite. Currently, the suite includes two main pipelines:

- The first pipeline is dedicated to genomic quality control and is encapsulated within the ideal-genom-qc package (Figure 1A).
- The second pipeline focuses on GWAS analysis and is built around the ideal-genom library (Figure 1B).

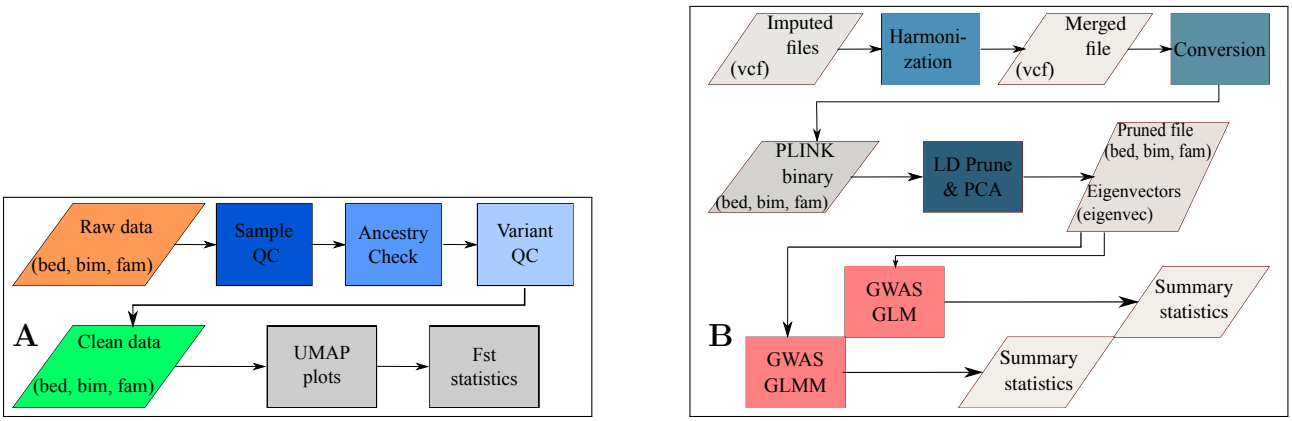

Figure 1 — **Panel A:** Workflow of the QC pipeline. **Panel B:** Workflow of the GWAS pipeline.

Each pipeline is composed of smaller sub-pipelines, which facilitates maintenance and further development. This modular approach also allows users to customize specific steps according to their needs and expertise. Additionally, users can execute individual components of the pipeline without running the entire workflow, making it easier to adjust the parameters efficiently.

Unlike GenoTools [15] and plinkQC [16] pipelines, which require all parameters to be fed through the CLI or a bash script, IDEAL-GENOM ensures complete reproducibility through centralized parameter management. Both workflows consolidate all critical configuration parameters into a

single `JSON` file (one for each pipeline), eliminating the error-prone process of introducing values by hand in the terminal. The comprehensive default configuration files are readily available in the `README.md` of the GitHub repository, allowing research teams to independently validate results with minimal effort and maximum consistency.

Among its many advantages, the `JSON` format allows users to manually edit configurations and validate them using external tools such as JSONLint or JSON Formatter.

The `ideal-genom-qc` and `ideal-genom` libraries are designed to be executed via a Python-based command-line interface, offering flexibility and reproducibility for genomic data quality control and GWAS pipeline. The QC pipeline is invoked using the syntax shown in Figure 2C:

The CLI accepts several mandatory arguments:

- `--path_params`: Path to a `JSON` file containing parameter configurations for the analysis.
- `--file_folders`: Path to a `JSON` file specifying input and output directory structures.
- `--steps`: Path to a `JSON` file defining the processing steps to be executed.
- `--recompute-merge`: Boolean flag indicating whether intermediate merged data should be recomputed during ancestry outlier analysis.
- `--build`: Genome build version (e.g., 37 or 38).

This interface enables full control over the pipeline execution, supporting modular configuration through structured `JSON` files. It is well-suited for integration into automated workflows or batch processing environments.

The CLI syntax for `ideal-genom` is identical, except the `--recompute-merge` argument is not required in this context.

3.1 Genomic QC pipeline

As previously mentioned, the QC pipeline is integrated within the `ideal-genom-qc` library and follows the guidelines outlined in [2]. The pipeline is designed to minimise human intervention, requiring only a few CLI parameters to be executed.

The hierarchically organized pipeline operates on a per-project basis, meaning that each cohort requires its own dedicated directory to ensure proper functionality. Within each project directory, a hierarchical structure of subdirectories is created, which organizes, and stores data files associated with the source dataset. This well-defined folder structure is essential for maintaining an organized workflow, as it allows for clear separation and easy retrieval of files related to each specific project. Moreover, this systematized hierarchical organization enhances data traceability, enabling users to track the provenance of each file and its role within the pipeline. By adhering to this structure, the pipeline not only facilitates smoother data management but also supports reproducibility, ensuring that the entire analysis process can be accurately reconstructed and verified. See Figure 2D.

The QC pipeline consists of three core steps, along with two optional steps:

- Sample QC,
- Ancestry check,
- Variant QC,
- (optional) Population structure analysis using UMAP,

```
{                                   A
  "sample": true,
  "ancestry": true,
  "variant": true,
  "umap": false,
  "fst": false
}
```

```
{                                   B
  "sample": {...},
  "ancestry": {...},
  "variant": {...},
  "umap": {...},
  "fst": {...}
}
```

```
$ python ideal_genom_qc \                C
  --path-params <path_to_params.JSON> \
  --steps <path_to_steps.JSON> \
  --file-folder <path_to_paths.JSON> \
  --recompute-merge true \
  --build 38
```

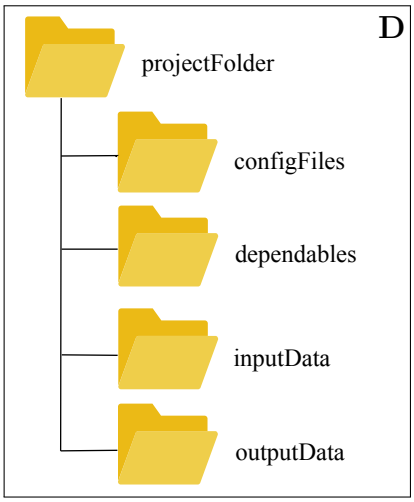

Figure 2 — **Panel A.** Example `JSON` configuration specifying which processing steps to execute. **Panel B.** Example `JSON` configuration showing the nested structure with the parameters required for each step. **C.** Command-line call with its corresponding parameters. **D.** Folder structure of the project.

- (optional) Computation of Fst statistics with respect to 1000G super populations

While the first three steps are standard in genomic QC, the optional population structure analysis step provides an enhanced, comprehensive visualization of genetic diversity. This step leverages the `umap-learn` library to capture non-linear structures within the dataset. Further details on this aspect are provided below.

The execution of the pipeline's core components is controlled by the `steps.JSON` file, which follows the structure outlined Figure 2A.

When a key is set to true, the corresponding step is executed as part of the QC pipeline. Conversely, if a key is set to false, that step will be skipped. Since each step depends on the output of the previous one, missing required files will throw an exception.

One of the key advantages of this configuration file is that users can modify parameters and control the execution of each step without the need to rerun the entire pipeline. This flexibility is particularly useful in iterative workflows, where only certain steps require adjustment or refinement. By setting the respective keys to true or false, users can selectively enable or disable specific steps, allowing for targeted quality control or analysis. Moreover, this structure enables users to focus on individual steps, providing the option to inspect and verify the output of each step before proceeding to the next step. As a result, users gain more control over the process, which enhances efficiency and minimises the time spent debugging or reprocessing unnecessary steps. This modular approach not only improves workflow optimization but also facilitates reproducibility, as users can document and execute specific parts of the pipeline as needed.

To facilitate reproducibility of the entire QC process, all parameters are stored in a file named `parameters.JSON`. Given the number of required parameters, we recommend checking its structure in the `README.md` file on GitHub. A detailed description of the parameters for each step, along with their default values, is provided in Table 1; see also Figure 2B.

To better understand the functionalities of each pipeline component, we now provide a detailed breakdown of its key steps.

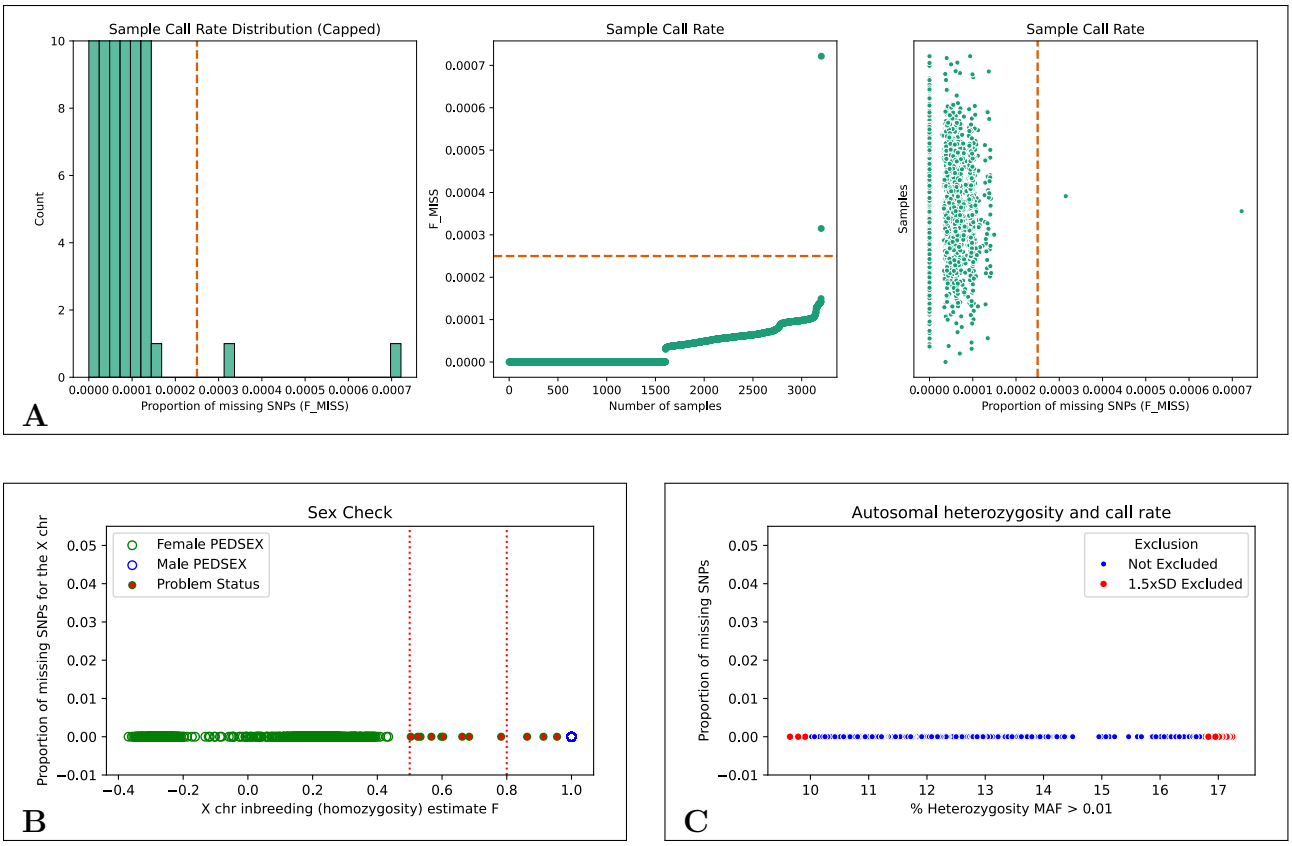

Figure 3 — **Panel A.** Dashboard showing a histogram of the sample call rate, and the number of samples that will be dropped according to the threshold considered. **Panel B.** Plot of the X chromosome inbreeding estimate versus the ratio of missing SNPs, that might help to determine mislabeled individuals. **Panel C.**. Plot of the X chromosome inbreeding estimate versus the ratio of missing SNPs.

In the sample quality control step, three Boolean parameters control the general behavior of the pipeline: `rename_snp`, `hh_to_missing`, and `use_kingship`. When `rename_snp` is set to true, SNP identifiers are reformatted to the structure chr:pos:a2:a1, a workaround implemented to avoid issues caused by overly long SNP names that can be problematic for PLINK. The `hh_to_missing` parameter, if enabled, will convert heterozygous haploid hardcalls and all female calls on chromosome Y to missing values. Lastly, `use_kingship` determines the method used to identify closely related samples — either identity by descent (IBD) or the KING-robust algorithm [17].

Since all cohorts have their own peculiarities, the parameters of a quality control pipeline should not be considered static, but they should be chosen with respect to the characteristics of the population at hand. Thus, we have designed a set of plots to visualize the outcome of the quality control pipeline and with them, the user will have more available information to choose if it keeps the default parameters or if it requires some variation. We have run a test of the QC pipeline using 1000G data, and to obtain meaningful plots we had to choose "unusual" parameters given the high quality of the 1000G samples.

When performing QC for 1000G data, plots showed that the default threshold for dropping samples according to the call rate is quite high, so it can be lowered without losing too many samples (Figure 3A). Moreover, the plot we have designed to report the sex check step also shows

that many females have a high value for the F statistic, if we had kept the default value many samples would be lost, but the plot allows to visually and empirically determine a better threshold for determining the problematic samples (Figure 3B).

Since the autosomal heterozygosity check has been divided according to MAF, on one hand the analysis was done for markers with MAF greater than 0.01 and on the other hand for those SNPs with MAF below 0.01, we draw a plot for each part of the analysis, though both follow the same design. Firstly, a twin histogram is plotted, one showing the full bin distribution and another with the upper limit capped to make a detailed picture of the distribution visible. This is complemented with a plot of the percent of heterozygosity versus proportion of missing SNPs, (Figure 3C) where the samples with a given deviation from the mean are highlighted. For 1000G data we had to use a threshold of 1.5 standard deviations to obtain highlighted samples, while the default value is 3 standard deviations.

For the ancestry check, we use the 1000G as our reference panel. Our pipeline includes functionality to automatically download the reference panel (GRCh38 build by default) and process it to generate the binary files required for ancestry analysis. During implementation, we identified challenges with duplicate rsIDs within the 1000G dataset. To address this issue, our pipeline automatically standardizes SNP identifiers by renaming them to the format chr:pos:a2:a1. This is another reason to rename the SNPs at sample QC start, ensuring consistent SNP identification across datasets. This approach is critical because the merging of the reference panel with the study cohort data requires exact matching of SNP identifiers, because any discrepancy would cause the merge operation to fail, compromising the ancestry check.

Our approach to detecting population outliers is simple and conservative:

1. Data alignment: The study dataset is first aligned with the reference panel through preparatory steps.

2. Principal Component Analysis (PCA): The pipeline computes the principal component decomposition of the dataset resulting from the merging of the study data and reference panel.

3. Outlier detection:

   - Samples deviating by more than `ancestry_qc['ref_threshold']` standard deviations from the mean of the reference population are flagged as potential outliers.
   - Samples deviating by more than `ancestry_qc['stu_threshold']` standard deviations from the mean of the study population are also flagged.

4. The final set of population outliers consists of the intersection of both groups.

Users can adjust these thresholds to obtain the best results for their cohort. The ancestry check is complemented by a 2D and 3D PCA plot, allowing for (simple) visual exploration of population structure. In its current form, the pipeline only allows to check the population structure against one of the Super Populations present in the 1000G data. We have planned to enrich this part of the pipeline in a future release.

For variant QC, there are some filtering steps that are done "explicitly", and others are done "implicitly". That is, the call rate for males and females, the check of different call rates between cases and controls as well as the Hardy-Weinberg equilibrium have each one its dedicated step, while the filtering by minor allele frequencies and call rate among all samples are done when generating the clean files. Each one of these explicit steps has a companion histogram plot, so the user can analyse the threshold that best fits the data at hand.

As previously mentioned, the fourth step of the pipeline is optional—but highly recommended—as it delivers a deeper, refined understanding of the cohort. This step leverages the UMAP algorithm, a powerful tool that reveals nonlinear patterns in population structure that are often invisible to traditional linear methods like PCA. While PCA reduces dimensionality through linear projections, UMAP excels by preserving both local and global relationships within the data, generating a rich, nuanced visualization of genetic diversity and population stratification. This added layer of insight is crucial for detecting subtle structures and hidden subpopulations. To our knowledge, this is the first integration of UMAP into a genomic quality control pipeline, providing a novel and visually intuitive perspective on cohort composition [18, 19]. This innovation positions our tool not just as a QC step, but as a gateway to deeper population insights.

Before performing the UMAP projection, the cleaned dataset (post-QC) is pruned to avoid misleading visualizations, as recommended in previous work [18]. The pipeline allows users to adjust UMAP's three main parameters—`min_dist`, `n_neighbors`, and `metric`—directly via the configuration file. For detailed explanations of these parameters, we refer users to the official UMAP documentation [20].

A key advantage of our implementation is that it supports simultaneous exploration of multiple parameter combinations. Users can specify lists of values for each parameter in the configuration file, allowing the pipeline to construct a parameter grid. For each combination, a two-dimensional UMAP projection is computed, and a corresponding scatter plot is generated (Figure 4B) In addition, also a PCA plot is generated (Figure 4A), so the user can compare the outcome of the UMAP dimensionality reduction and PCA. With the 1000G dataset, UMAP discovers more variety than PCA.

It is important to note that the UMAP algorithm introduces stochasticity; therefore, to ensure full reproducibility, users should set a fixed value for the `random_state` parameter.

The resulting plots can be customized in two main ways: (1) by specifying different markers for case and control groups, and (2) by adding color coding (hue) based on a user-defined categorical variable. If no hue variable is provided, the case-control status (as read from the `fam` file) is used by default. The distribution of categories for the color coding must be supplied in an additional tab separated text file.

3.2 GWAS pipeline

Let us recall that the GWAS pipeline is part of the `ideal-genom` library. The design principles that led us during its development are the same as we followed for the QC pipeline. Therefore, the folder structure is like the one shown in Figure 2D. Moreover, we have designed the process flow as a "forked" pipe, then it can be viewed as a standard wye pipe. We have decomposed the workflow into four steps, with the goal of a more modular and versatile functioning that could also be integrated into different processes, the `steps.JSON` has the same structure as the corresponding file from the QC pipeline. In this case the steps, that is the keys of the `steps.JSON` file, are `post_imputation`, `preparatory`, `gwas_glm` and `gwas_glmm`.

The `JSON` file with the parameters needed to run the pipeline is very similar to the one needed to execute the QC pipeline. For further details, the default parameters can be obtained from the `README.md` file on GitHub.

This pipeline is designed to accept input (zipped and password protected) `vcf` files, which are the typical files used as output of genetic imputation software. Then, the first step of our GWAS pipeline is to process "freshly" imputed data. During this step, the imputed variants are filtered according to R^2 value, and then the files are normalized and annotated (with rsIDs), and then vcf files are converted to PLINK binary files and the fam files are updated with sex and phenotype.

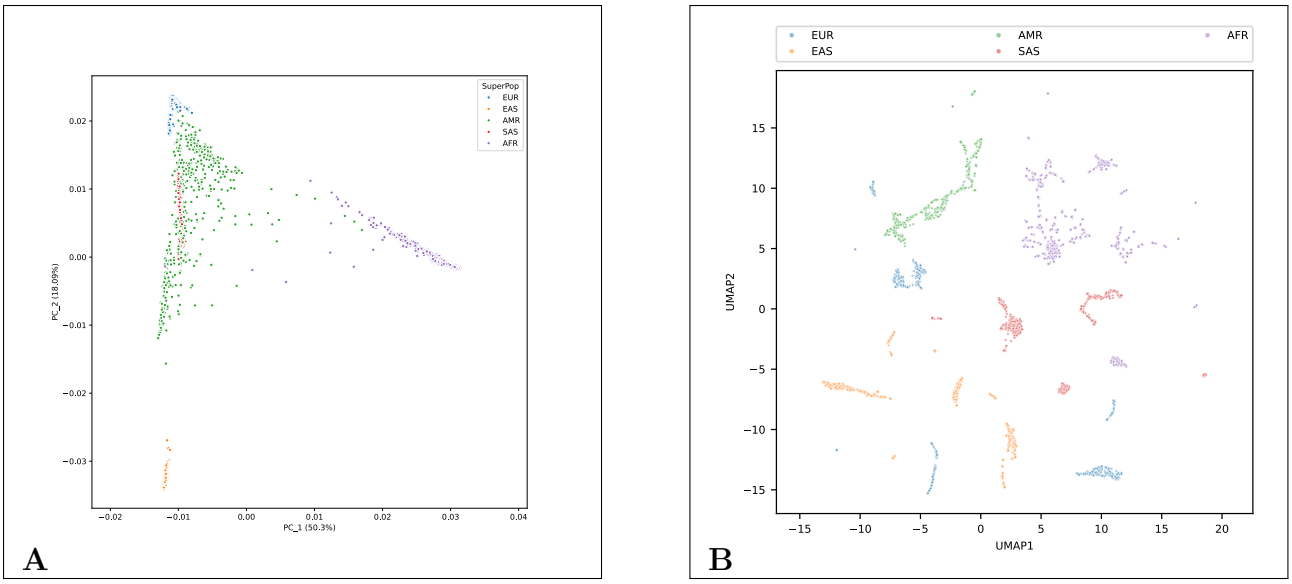

Figure 4 — **Panel A.** Plot showing the PCA decomposition of the 1000G cohort. **Panel B.** Plot drawn after the UMAP projection of the same cohort with `n_neighbors` equal 5, `min_dist` equal 0.2 and `metric` equal to 'euclidean'.

The functions leverage Python functionalities to use computing resources efficiently and perform as many parallel processes as possible. This stage requires one external file that must be in the dependable directory: the reference genome (in FASTA file) to normalize the data, and a vcf file to annotate the variants with rsIDs. In the current version, only the vcf must be provided by the user, since the FASTA file could be fetched from the web automatically and reused for another projects. The second stage is rather short, and its goal is to obtain the principal component decomposition that will be used as a covariate during the association analysis. It also constitutes the point where the workflow bifurcates and will flow along two branches, one for the GLM and another for the GLMM.

Both branches have a similar structure; the main difference lies in that the GLMM model requires a few more steps and it relies on GCTA for its computation, while for the GLM we use PLINK2.0. We must remark that GCTA software is memory intensive, making this option not suitable to run on local machines but in cloud-based machines with high RAM volume. Once the summary statistics are ready, a stepwise model selection of independently associated SNPs is run with GCTA to obtain top hits among the significative SNPs. In addition, we have included an additional step to automatically annotate with the nearest gene names the top SNPs. We need to point out that our suite has an advantage over GWASlab [21]: the library can dynamically fetch the latest release of the two most well-known databases to perform gene annotations: NCBI and Ensembl. Gene annotation can be performed using build GRCh37 or GRCh38 according to the user's needs. The modular design of the toolkit also allows the usage of the annotation tool outside the GWAS pipeline.

3.3 Visualization capabilities

Generating GWAS summary statistics is only the beginning—researchers must further analyze the results to extract meaningful insights. Given the size and complexity of GWAS data, effective visu-

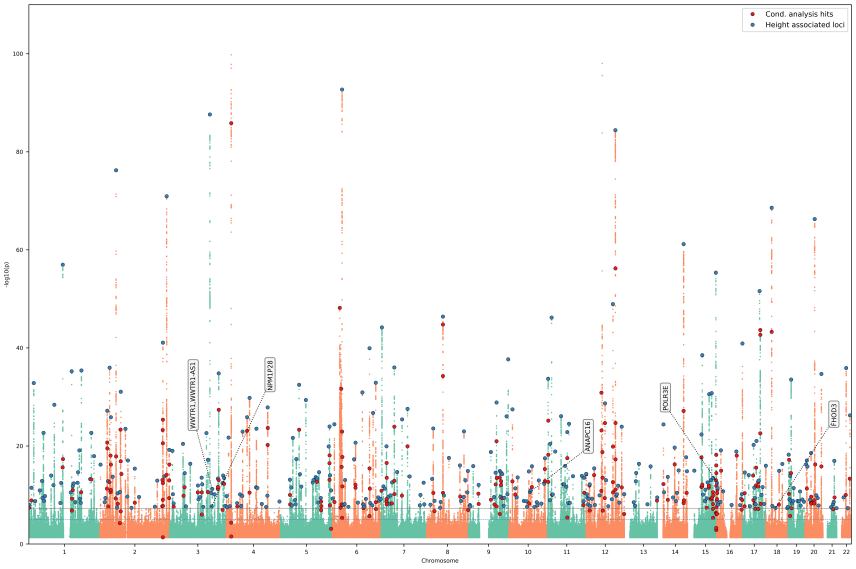

Figure 5 — Manhattan plot draw with data from [24]. There are two types of highlighted SNPs as well as some with the nearest gene annotation.

alization is crucial for interpretation and pattern recognition. The IDEAL-GENOM suite provides a comprehensive set of visualizations commonly used in GWAS analysis, including Manhattan and QQ plots. Additionally, it offers comparative visualizations such as Miami plots and effect-size comparison plots, enabling researchers to contrast findings across studies.

Since GWAS plots often involve numerous parameters and require fine-tuning (e.g., adjusting colors, sizes, and significance thresholds), we have separated the plotting functions from the main analysis pipelines. To enhance usability, we have also developed dedicated example notebooks for each available plot, allowing users to quickly generate and customize visualizations as needed. To build the example notebooks, real data is fetched automatically from well-known databases such as GWAS Catalog. Most of the plotting functionalities allow highlighting of SNPs of interest if a list of identifiers is provided by the user and also can be annotated with user-provided gene names or automatically relying on the library capabilities. Regarding this last point, the user can choose between build GRCh37 or GRCh38 of the human genome, according to his needs as well as choosing the database of his preference between Ensembl or NCBI.

3.3.1 Manhattan plot

Manhattan plots are among the most widely used visualizations in GWAS research [22, 23]. Our implementation balances visual appeal with ease of use. To avoid excessive colorization, we use a simple two-color scheme that maintains clarity while highlighting key findings. Our implementation allows us to highlight groups of SNPs with different colors in case different sets of characteristics want to be emphasized. Even though there are many R packages and Python libraries that can draw Manhattan plots with multiple functionalities (a quick search in GitHub alone gives more than 100 repositories), most of them are stand-alone applications, and it is difficult to know which one is kept up to date, what can be a burden for the maintenance of our own toolkit. For that reason, we decided to develop our version. See Figure 5.

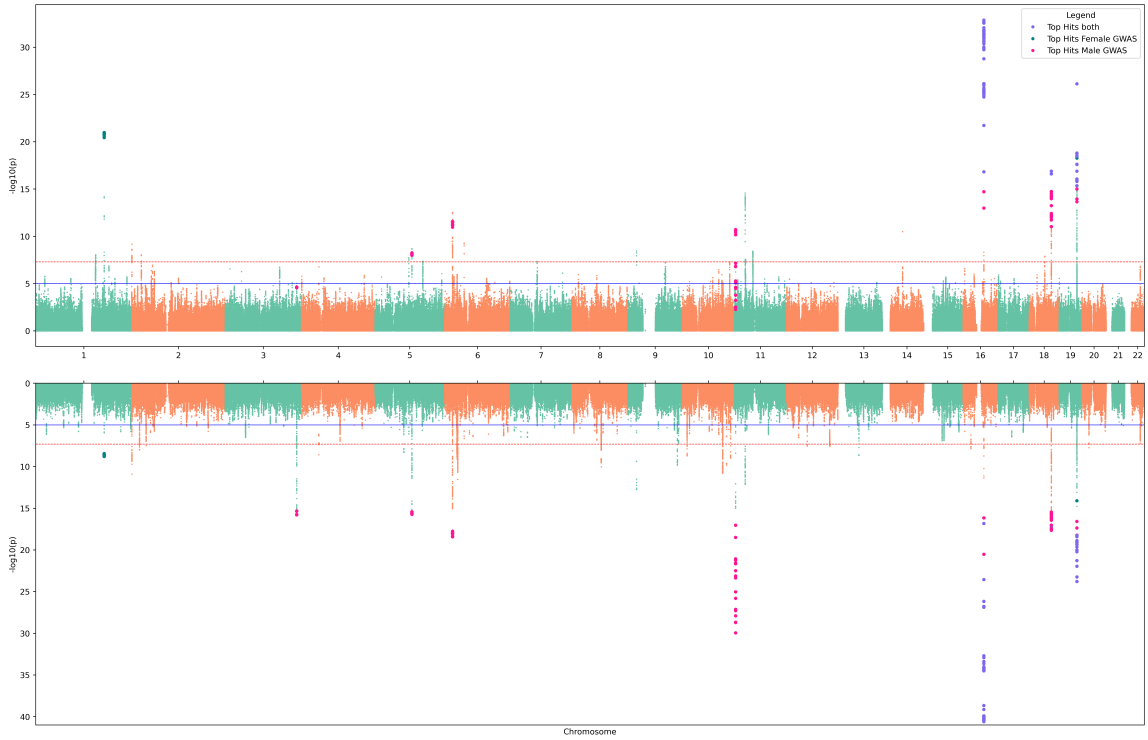

Figure 6 — Miami plot generated with data from [25]. Highlighted SNPs were selected by the authors according to the ones with lower p-values. For simplicity there us no annotations in this plot.

3.3.2 Miami plot

Miami plots have become an essential tool to compare two studies and visualize the summary statistics differences, and it takes its name from the resemblance of the mirror image of a skyline on the sea, like what one can get in Miami harbor. Our function allows us to highlight different groups of SNPs on the bottom and top plots, and it adds the functionality to a different highlight for those SNPs that coincide in the highlighting. Moreover, gene annotations can be done on both plots independently, giving more freedom to the user. See Figure 6.

3.3.3 QQ plot

Another classical visualization among genomic research community is QQ plot (quantile-quantile plot) and it is used in GWAS to check if the distribution of p-values follows what we expect under the null hypothesis (no true associations). It compares the observed p-values (y-axis) to the expected p-values (x-axis) under a uniform distribution. We enriched the plot with the computation of the genomic inflation factor (λ). See Figure 7B.

3.3.4 Beta vs beta plots

Another useful plot to compare studies is the beta-vs-beta plot, also known as beta-beta plot. This kind of plot is useful to compare the effect size (β) of a linear model used to perform a GWAS. The plot helps to visualize if the effects are consistent across the two studies, whether there's a positive or negative correlation, and if there are any major discrepancies. Our implementation was deeply inspired by [21]. See Figure 7A.

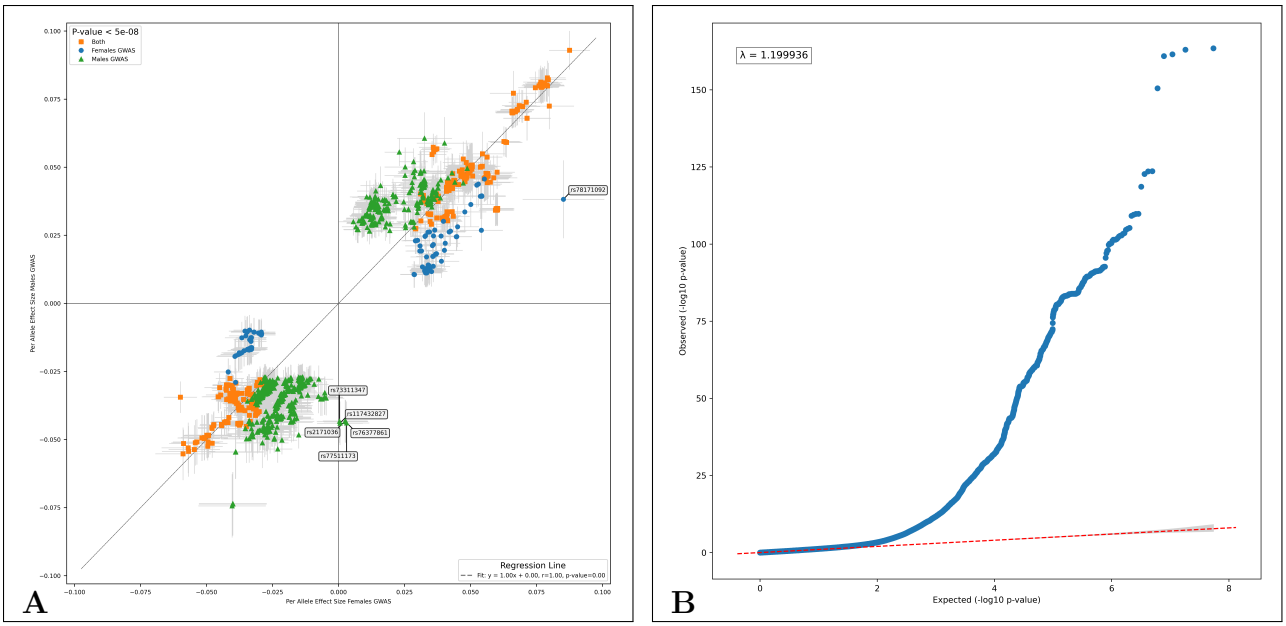

Figure 7 — **Panel A.** Effect size comparison plot drew with data from [25]. **Panel B.** QQ-plot drew with data from [24].

3.3.5 Trumpet plot

Trumpet plots were introduced in 2023 to set a standard to visualize the effect size of risk variants across the allele frequency spectrum [26]. In our toolkit, we have developed a function to draw trumpet plots for binary and quantitative traits. Moreover, we have left it to the user the decision to use the logarithmic or linear scale for the allele frequency axis, instead of restricting it to only the logarithmic. An example can be seen in Figure 8. Up to the moment this manuscript is written, the authors do not know any Python library other than GWASlab that has this functionality.

3.4 Validation of IDEAL-GENOM results

An essential part of developing a new toolkit for high-quality scientific analysis is validating the results obtained against established datasets. In our case, ideal-genom was developed alongside a major research project in our lab, which investigated a large Indian cohort to identify SNPs associated with Parkinson's disease. As a result, we were able to validate the quality control procedures performed by our toolkit against those required for the aforementioned study, as well as the GWAS summary statistics [27]. Additionally, the visualization capabilities of our toolkit were tested against previously published studies [25, 24]. These validations, along with detailed examples, can be scrutinized in the example notebooks included as part of our toolkit.

Table 1 — Parameters and their default values for the QC pipeline.

Parameter	Description	Type	Default Value
Sample QC			
rename_snp	Rename SNP to chr:pos:a2:a1	bool	true
hh_to_missing	Set hh warnings to missing	bool	true
use_kingship	Use kingships for duplicates	bool	true

Parameter	Description	Type	Default Value
ind_pair	Independence pairwise	list	[50, 5, 0.2]
mind		float	0.2
sex_check		list	[0.2, 0.8]
maf	Minor allele frequency	float	0.01
het_deviation		float	3
kingship	Threshold to filter samples out by kingship	float	0.354
ibd_threshold	Threshold to filter samples out by IBD	float	0.185
Ancestry Check			
ind_pair	Independence pairwise	list	[50, 5, 0.2]
pca	Number of principal components	integer	10
maf	Minor allele frequency	float	0.01
ref_threshold	Number of deviations from the mean of the reference population	float	4
stu_threshold	Number of deviations from the mean of the study population	float	4
reference_pop	Identifier of the reference population in 1KG	string	"SAS"
num_pcs	Number of principal components to use when flagging outliers	integer	10
Variant QC			
chr_y	Identifier of the Y chromosome	integer	24
miss_data_rate		float	0.2
diff_genotype_rate		float	1e-5
geno		float	0.1
maf	Minor allele frequency	float	5e-8
hwe	Hardy-Weinberg equilibrium	float	5e-8
UMAP			
umap_maf	Minor allele frequency	float	0.01
umap_mind		float	0.2
umap_hwe	Hardy-Weinberg equilibrium	float	5e-8
umap_geno		float	0.2
umap_ind-pair	Independence pairwise	list	[50, 5, 0.2]
umap_pca	Number of principal components to feed the UMAP algorithm	integer	10
n_neighbors	Numbers of neighbors	list	[5]
metric	Metric used by UMAP	list	["euclidean"]
min_dist	Minimal distance used by UMAP	list	[0.01]
random_state	Random state for UMAP algorithm	integer	42
case_control_marke	Use different markers for cases and controls	bool	true
color_hue_file	Path to file with categories	string	path/to/color_hue_file.txt
umap_kwargs	Other parameters for UMAP algorithm	dictionary	{}

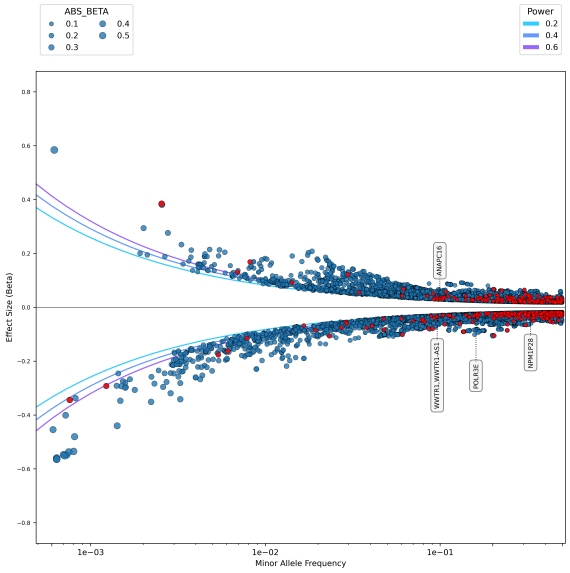

Figure 8 — Trumpet plot drew with data fetched from [24].

	GenoTools	**GWASLab**	**plinkQC**	**IDEAL-GENOM**
Programming Language	Python3	Python3	R	Python3
QC pipeline	yes	no	yes	yes
QC plots	no	no	yes	yes
GWAS	yes	no	no	yes
QQ plots	yes	yes	no	yes
Manhattan plot	no	yes	no	yes
Miami plot	no	yes	no	yes
Effect size comparison	no	yes	no	yes
Zoom plot	no	no	no	yes
Trumpet plot	no	yes	no	yes

Table 2 — Comparison of features across different software.

4 Discussion

IDEAL-GENOM toolkit provides an efficient approach for handling large-scale genetic datasets to provide an efficient platform to perform genomic analysis. Designed for scalability and flexibility, the toolkit has also been successfully adopted by external research teams to streamline their population genetics analyses.

The core strength of IDEAL-GENOM lies in its ability to condense complex, multi-step analytical workflows into a modular and user-customizable command-line interface. Its robust quality control pipeline includes high-resolution visualizations and integrates dimensionality reduction techniques such as UMAP for assessing genetic structure. Furthermore, IDEAL-GENOM supports comprehensive GWAS workflows and allows the generation of publication-ready visual outputs from summary statistics, significantly accelerating the path from raw data to results.

5 Conclusions

In conclusion, IDEAL-GENOM is designed to improve the efficiency and reproducibility of GWAS-based studies by offering modular and customizable workflows and producing publication-quality figures. Its flexibility and user-centric design make it an asset for the scientific community engaged in GWAS analysis.

IDEAL-GENOM offers a wide range of capabilities and customization to beginners and advanced users to perform GWAS. Moreover, IDEAL-GENOM offers Jupyter notebooks as a tutorial to teach intricacies of GWAS analysis for undergraduate and doctoral fellows.

In Table 2 a comprehensive comparison of our toolkit and different software with similar goals is provided, although there is no full overlap with neither of them.

References

[1] R. J. Loos. "15 years of genome-wide association studies and no signs of slowing down". In: *Nature communications* 11.1 (2020), p. 5900. DOI: https//doi.org/10.1038/s41467-020-19653-5.

[2] C. A. Anderson, F. H. Pettersson, G. M. Clarke, L. R. Cardon, A. P. Morris, and K. T. Zondervan. "Data quality control in genetic case-control association studies". In: *Nature protocols* 5.9 (2010), pp. 1564–1573. DOI: https://doi.org/10.1038/nprot.2010.116.

[3] S. Turner, L. L. Armstrong, Y. Bradford, C. S. Carlson, D. C. Crawford, A. T. Crenshaw, et al. "Quality control procedures for genome-wide association studies". In: *Current protocols in human genetics* 68.1 (2011), pp. 1–19. DOI: https://doi.org/10.1002/0471142905.hg0119s68.

[4] T. W. T. C. C. Consortium. "Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls". In: *Nature* 447.7145 (2007), pp. 661–678. DOI: https://doi.org/10.1038/nature05911.

[5] P. M. Visscher, M. A. Brown, M. I. McCarthy, and J. Yang. "Five years of GWAS discovery". In: *The American Journal of Human Genetics* 90.1 (2012), pp. 7–24. DOI: https://doi.org/10.1016/j.ajhg.2011.11.029.

[6] P. M. Visscher, N. R. Wray, Q. Zhang, P. Sklar, M. I. McCarthy, M. A. Brown, et al. "10 years of GWAS discovery: biology, function, and translation". In: *The American Journal of Human Genetics* 101.1 (2017), pp. 5–22. DOI: https://doi.org/10.1016/j.ajhg.2017.06.005.

[7] A. Abdellaoui, L. Yengo, K. J. Verweij, and P. M. Visscher. "15 years of GWAS discovery: realizing the promise". In: *The American Journal of Human Genetics* 110.2 (2023), pp. 179–194. DOI: https://doi.org/10.1016/j.ajhg.2023.01.006.

[8] S. Purcell, B. Neale, K. Todd-Brown, L. Thomas, M. A. Ferreira, D. Bender, et al. "PLINK: a tool set for whole-genome association and population-based linkage analyses". In: *The American journal of human genetics* 81.3 (2007), pp. 559–575. DOI: https://doi.org/10.1086/519795.

[9] C. C. Chang, C. C. Chow, L. C. Tellier, S. Vattikuti, S. M. Purcell, and J. J. Lee. "Second-generation PLINK: rising to the challenge of larger and richer datasets". In: *Gigascience* 4.1 (2015), s13742–015. DOI: https://doi.org/10.1186/s13742-015-0047-8.

[10] S. Purcell and C. Chang. *PLINK2.0 home*. 2024. URL: https://www.cog-genomics.org/plink/2.0/..

[11] J. Yang, S. H. Lee, M. E. Goddard, and P. M. Visscher. "GCTA: a tool for genome-wide complex trait analysis". In: *The American Journal of Human Genetics* 88.1 (2011), pp. 76–82. DOI: https://doi.org/10.1016/j.ajhg.2010.11.011.

[12] J. Healy and L. McInnes. "Uniform manifold approximation and projection". In: *Nature Reviews Methods Primers* 4.1 (2024), p. 82. DOI: https://doi.org/10.1038/s43586-024-00199-y.

[13] 1. G. P. Consortium et al. "A global reference for human genetic variation". In: *Nature* 526.7571 (2015), p. 68.

[14] P. Danecek, J. K. Bonfield, J. Liddle, J. Marshall, V. Ohan, M. O. Pollard, et al. "Twelve years of SAMtools and BCFtools". In: *Gigascience* 10.2 (2021), giab008. DOI: https//doi.org//10.1093/gigascience/giab008.

[15] D. Vitale, M. J. Koretsky, N. Kuznetsov, S. Hong, J. Martin, M. James, et al. "GenoTools: An open-source Python package for efficient genotype data quality control and analysis". In: *G3: Genes, Genomes, Genetics* 15.1 (2025), jkae268.

[16] H. V. Meyer. *meyer-lab-cshl/plinkQC: plinkQC version 0.3.4*. Version v0.3.4. July 2021. DOI: 10.5281/zenodo.5106689. URL: https://zenodo.org/records/5106689.

[17] A. Manichaikul, J. C. Mychaleckyj, S. S. Rich, K. Daly, M. Sale, and W.-M. Chen. "Robust relationship inference in genome-wide association studies". In: *Bioinformatics* 26.22 (2010), pp. 2867–2873. DOI: doi.org/10.1093/bioinformatics/btq559.

[18] A. Diaz-Papkovich, L. Anderson-Trocmé, and S. Gravel. "A review of UMAP in population genetics". In: *Journal of Human Genetics* 66.1 (2021), pp. 85–91. DOI: doi.org/10.1038/s10038-020-0811-3.

[19] A. Diaz-Papkovich, L. Anderson-Trocmé, C. Ben-Eghan, and S. Gravel. "UMAP reveals cryptic population structure and phenotype heterogeneity in large genomic cohorts". In: *PLoS genetics* 15.11 (2019), e1008432. DOI: doi.org/10.1371/journal.pgen.1008432.

[20] L. McInnes. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. https://umap-learn.readthedocs.io/en/latest/. Accessed on 17 June 2025. 2025.

[21] Y. He, M. Koido, Y. Shimmori, and Y. Kamatani. *GWASLab: A Python package for processing and visualizing GWAS summary statistics*. https://jxiv.jst.go.jp/index.php/jxiv/preprint/view/370/. Preprint. 2023. DOI: doi.org/10.51094/jxiv.370.

[22] T. A. Pearson and T. A. Manolio. "How to interpret a genome-wide association study". In: *JAMA* 299.11 (2008), pp. 1335–1344. DOI: https://doi.org/10.1001/jama.299.11.1335.

[23] J. Hardy and A. Singleton. "Genomewide association studies and human disease". In: *New England Journal of Medicine* 360.17 (2009), pp. 1759–1768. DOI: doi.org/10.1056/NEJMra0808700.

[24] M. Akiyama, K. Ishigaki, S. Sakaue, Y. Momozawa, M. Horikoshi, M. Hirata, et al. "Characterizing rare and low-frequency height-associated variants in the Japanese population". In: *Nature communications* 10.1 (2019), p. 4393. DOI: doi.org/10.1038/s41467-019-12276-5.

[25] M. Akiyama, Y. Okada, M. Kanai, A. Takahashi, Y. Momozawa, M. Ikeda, et al. "Genome-wide association study identifies 112 new loci for body mass index in the Japanese population". In: *Nature genetics* 49.10 (2017), pp. 1458–1467. DOI: doi.org/10.1038/ng.3951.

[26] L. Corte, L. Liou, P. F. O'Reilly, and J. García-González. "Trumpet plots: visualizing the relationship between allele frequency and effect size in genetic association studies". In: *GigaByte* 2023 (2023), gigabyte89. DOI: doi.org/10.46471/gigabyte.89.

[27] A. Kishore, A. Ashok Kumar Sreelatha, A. M. Tenghe, R. Borgohain, D. K. Puthanveedu, R. Rajan, et al. "Deciphering the Genetic Architecture of Parkinson's Disease in India". In: *medRxiv* (2025), pp. 2025–02. DOI: doi.org/10.1101/2025.02.17.25322132.

Declarations

Ethics approval and consent to participate

Not applicable

Consent for publication

Not applicable

Availability and Requirements

Project name: IDEAL-GENOM
Project home page: https://ideal-genom-qc.readthedocs.io/en/latest/ and https://ideal-genom.readthedocs.io/en/latest/.
Operating system(s): Platform independent
Programming language: Python
Other requirements: pandas, matplotlib, seaborn, umap-learn, scipy
License: MIT
Any restrictions to use by non-academics: no restrictions.

Availability of data and material

No data was generated during this research. The data used is freely available on 1000 Genome Project website: https://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/ and PLINK2 official website https://www.cog-genomics.org/plink/2.0/resources.

Competing interests

The authors declare no competing interests.

Funding

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) research grant GZ: SH 599/16-1.

Authors' contributions

LGGR wrote the main text of the paper and developed the software. MS made important revisions and improvements to the main text. ASKS and AMMT designed the structure of the pipelines. All authors reviewed the manuscript.

Acknowledgements

We want to acknowledge the valuable feedback given by Paola di Martino as well as suggestions for the improvement of software.