DOCUMENTATION REPORT

Arch Team
Cheau Lee, Farzard Hamzawe, Roxana-Andreea Bosnea, Siril Sunil

Msc Bioinformatics
Bioinformatics Project – BIO727P

Table of Contents

1. Introduction

Developed by Team Arch, AG-ArchGenome is a web application engineered to equip users with comprehensive analytical tools for elucidating population structure through the examination of genetic data from diverse populations, as sourced from Variant Call Format (VCF) files. The platform integrates a suite of advanced analytical methods, including Principal Component Analysis (PCA), Admixture Analysis, and Population Genetic Differentiation, to give researchers and geneticists a robust mechanism for a detailed population genetic study.

The concept of population structure is important in genetic research, where it significantly influences the interpretation of genetic variation data. It is of paramount importance across a spectrum of disciplines such as *evolutionary genetics, conservation genetics, and human genetics*. Fundamentally, population structure delineates the extent of genetic homogeneity or heterogeneity among subdivisions within a collective dataset. Variability in genetic composition often arises from distinct factors predominantly the geographical distribution of sampled groups, which may be isolated or dispersed along a geographical continuum. A profound comprehension of the population structure is indispensable for the preliminary assessment of the dataset, thereby facilitating informed subsequent genetic analyses (Dutheil, 2021).

Notably, the application leverages admixture proportion inference, a methodological approach that shows the demographic and genetic mosaic of populations. This technique is involved in depicting the admixture proportions, offering a quantifiable measure of genetic mixing, and applying insights into the population's genetic ancestry and evolutionary dynamics (Rosenberg, et al., 2005)

Additionally, the application is adept at conducting pairwise population genetic differentiation analyses, a quintessential tool in the field of population genetics. This approach is dedicated to quantifying the genetic variance between and within populations, thereby enabling the examination of genetic divergence. It can help users infer if there has been significant or little gene flow between populations based on FST values which leads to the enrichment of the user's understanding regarding the population's genetic history and its evolutionary underpinnings (Wright, 1965).

2. Software Architecture

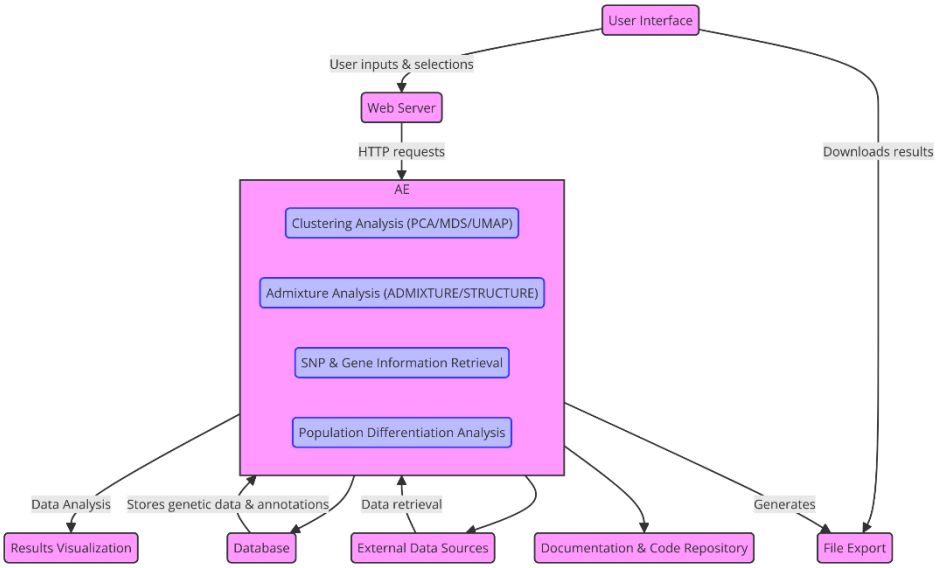

Figure 1. *This depicts a software architecture diagram/flowchart that was used for this project. AE stands for Analysis Engine.*

The construction of the flowchart was an integral initial phase in the development of the project, depicting the comprehensive architecture and functional interrelations within the system. The user interface (UI) serves as the principal point of interaction, where users can input data, select desired analytical procedures, and retrieve their results. Central to the system's architecture is the *Web Server*, functioning as an intermediary that processes HTTP requests from the UI, thereby orchestrating the subsequent analytical operations within the Analysis Engine (AE).

The AE constitutes the application's core, where various analytical processes are executed, and segmented into distinct modules:

- **Clustering Analysis Module**: This segment is dedicated to performing clustering analysis, aggregating genetic data based on inherent similarities. It offers the flexibility to select from a suite of dimensionality reduction techniques such as *Principal Component Analysis* (PCA), *Multidimensional Scaling* (MD), and *Uniform Manifold Approximation and Projection* (UMAP), each providing a unique approach to the visualization of high-dimensional genetic data.
- **Admixture Analysis Module**: Employed to deduce the population structure and allocate individuals to respective populations, this module suggests the utilization of tools like *ADMIXTURE* and *STRUCTURE*, which are extensively recognized for their efficacy in such analyses.
- **SNP & Gene Information Retrieval Module**: This function is responsible for the extraction of data pertaining to *single nucleotide polymorphisms* (SNPs) and genes, interfacing with a comprehensively populated database to fetch the requisite information.
- *Population Differentiation Analysis Module*: Focused on the examination of genetic disparities between populations, this module incorporates methodologies such as the *fixation index* (FST) to quantify population differentiation attributable to genetic structure.

Subsequent to the AE, the flowchart illustrates the *Database* component, which archives the genetic data and annotations procured by the AE for analytical processing. The inclusion of *External Data Sources* denotes the system's capability to integrate data from external genomic databases or repositories, such as NCBI or EMBL.

Additionally, the *Documentation & Code Repository* is depicted, highlighting the system's provision of documentation for both users and future developers, alongside a repository for the analytical codebase. The *File Export* underscores the system's ability to generate downloadable output files in *TXT* format. Conclusively, the *Results Visualisation* component is illustrated, highlighting the system's functionality to graphically represent the data and the analyses conducted, thereby providing users with intuitive and informative visual insights into their results.

2.1 Website Design

The web application's user interface is structured using *HTML* for underlying content, *CSS* for styling, and *JavaScript* for interactivity and responsiveness. All these web technologies contribute to the seamless and intuitive user experience.

The homepage consists of a simplistic design that features multiple boxes leading to different pages depending on the user's interest. It also displays the software name *AG - Arch Genome* at the header, and a *GitHub link* conveniently placed at the footer for ease of accessibility to the original repository for potential developers. One of the boxes is based on the *Abo'* page which leads the user to discover the project's objectives and overview of the features in the application to enhance understanding. This page also offers another link to the google drive containing the original *VCF file*, *annotation file*, and *database*. Thus, allowing the users to easily access relevant datasets. Additionally, it features the version of the application to maintain data integrity, should it be updated with newer genetic datasets in the future. The user can navigate back to the homepage to display the other options such as *SNP Information*, *Admixture Analysis*, *Clustering Analysis*, and *Population Differentiation Analysis* pages. All these lead to user selection pages before being submitted to the corresponding results page. Particularly for the *SNP Information Results* page, it also offers an extra notice for users to copy the displayed *CLNALLELEID* value and insert the value onto the *ClinVar* website for further updates. This idea facilitates the seamless integration between our website application and external resources for user flexibility based on their interests for further information.

BECAUSE OF FORMATTING ISUES, CHAPTER 3. DATABASE SCHEMA IS INCLUDED IN THE PAGE BELOW.

3. Database Schema

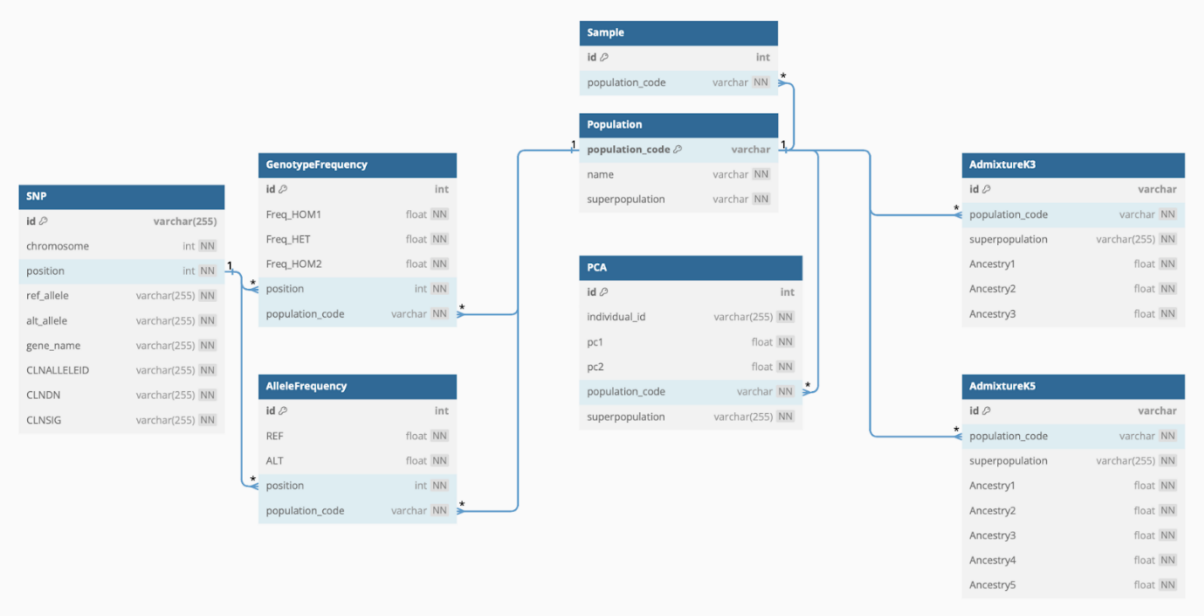

***Figure 2**. Diagram representation of the database schema. This displays individual tables along with highlighted interconnected relationships. These relationships are denoted by the symbols "1" (one-to-one relationship) and "*" (one-to-many relationship).*

The initial structure of the database schema comprises *SQLAlchemy*, an Object-Relational Mapping (ORM) library used for creating models that defines *tables*, *primary key constraints*, and *foreign key relationships*. This approach facilitates cross-referencing and data retrieval that is useful for analysis involving multiple tables. Likewise, these interactions enhance the code readability and maintainability. The Flask application is then configured with the *SQLite3* database, chosen for its lightweight nature and simplicity, in accordance with this project's requirements. Additionally, SQLAlchemy allows the switch of databases to other engines like *MySQL* and *PostgreSQL* giving developers the flexibility to scale up without changing too much code.

Figure 2 encompasses eight tables to store diverse genetic data and analysis results. Both *GenotypeFrequency* and *AlleleFrequency* tables are linked with the *SNP* table for retrieving genetic information, as well as corresponding data with Population Differentiation Analysis. The *PCA* table stores the results required for Clustering Analysis. Furthermore, the Admixture Analysis dataset is represented by two tables, *AdmixtureK3* and *AdmixtureK5*. However, it is important to note the *Sample* table, though included with a foreign key reference to *Population* table, is currently designed for potential use and is not actively used in the application. Future developers may choose not to populate this table if the tracking of individual samples is unnecessary, offering flexibility for further developments or specific use cases.

3.1 Data Collection

The dataset comprises genetic markers from *726 human samples*, collected from an unspecified location in Siberia, and integrated with whole-genome sequencing data from *26* other global populations sourced from the *1000 Genomes Project*, encompassing *3928 individuals* in total. The genetic information, specifically targeting *chromosome 1* (CHR1) for simplicity and efficiency, was acquired in a gzipped *Variant Call Format* (VCF) file. A supplementary annotation file was provided with each sample linked to its corresponding population code.

3.2 Sample Selection

For analytical purposes and to manage computational resources efficiently, a decision was made to **randomly** select a representative subset of **10 samples from each population** (using python_code_to_select_10indiv.py). The rationale behind selecting individuals at random is that the statistical principle of random sampling is more likely to be representative of the overall population. This method reduced the potential for selection bias that could skew the results of the analyses. Therefore, this approach helped to capture the genetic diversity present within each population while maintaining a manageable dataset that allows for efficient computation.

The inclusion of individual IDs in separate tables, such as *AdmixtureK3*, *AdmixtureK5*, and *PCA* becomes justified as it allows for cross-referencing results from different analyses and provides flexibility for querying. Developers should be aware of the interconnected nature of these tables and consider analytical requirements when handling or modifying individual IDs.

4. Analytical Implementation and Technologies

This section provides an in-depth evaluation of the software's development, demonstrating the implementation of each software requirement and the strategic selection of technologies to support the application's functionality. We delve into the core principles of the software's architecture, showing how certain features and capabilities were realised through careful planning, coding, and integration of multiple technological components. Our discussion extends to the technologies used, where we discuss the reasons for selecting each technology, emphasising its relevance and contribution to the project's overall objectives. This includes analysing the programming languages, frameworks, libraries, and other tools that contributed to the software's robust and efficient architecture.

In addition, we address the problems encountered during the software development cycle. Dealing with technology hurdles, optimisation issues, and other constraints that necessitated solutions or strategy shifts. Each part discussed how the obstacles for each analysis were found, as well as the strategies used to overcome them, giving a detailed picture of the project's problem-solving process.

4.1 Clustering Analysis

Clustering Analysis is an essential statistical tool for genetic data analysis as it classifies individuals into discrete genetic groups based on allelic frequencies. This stratification employs several algorithmic approaches, each with unique methodological strengths. We were given a choice to choose between *Principal Component Analysis* (PCA), *Multidimensional Scaling* (MDS), and *Uniform Manifold Approximation and Projection* (UMAP) based on data characteristics and research objectives (McInnes, et al., 2018).

Principal Component Analysis was chosen as the clustering method for the project because of its high ability to reduce dimensionality in genetic datasets. PCA is particularly effective at reducing high-dimensional data to principal components that capture the most important variance, revealing the dataset's genetic structure and stratification (Jolliffe, 2002). The following subchapter will go over why PCA is preferred over other methodologies.

Clustering analysis, particularly PCA, is valuable for more than just categorisation; it allows the creation of biplots that visually represent genetic distances and interactions between populations. These visual outputs are critical for identifying stratification and understanding the dataset's evolutionary and genetic narratives (Rosenberg, et al., 2002).

4.1.1 Technologies

Many Python modules were used in the PCA analysis, allowing for clear demonstrations of PCA execution and genetic data interpretation. The libraries were chosen for their durability, efficiency, and most importantly their usage in the scientific computing community.

Pandas (pd):
Pandas was used for data manipulation, specifically to arrange and prepare the genetic dataset for analysis, because its DataFrame structure can handle large datasets.

Scikit-learn:
Scikit-learn is a popular software overall and it was used in performing PCA and dimension reduction. It is useful for pattern recognition and data compression. Scikit-learn includes **scikit-allel (allel)**, which was also used to process and convert genetic variation into formats appropriate for PCA.

Seaborn & Matplotlib:
Seaborn and Matplotlib are visualisation packages that aid in the creation of visual graph representations, such as PCA results. Seaborn integrates seamlessly with Matplotlib and offers a more advanced interface for creating statistical graphics.

NumPy:
NumPy was used to perform numerical computations and transformations during pre-processing and PCA analysis.

4.1.2 Methodology

Selection Rationale for PCA:
After careful consideration of the strength and limits of each clustering method, the choice was taken to adopt Principal Component Analysis (PCA) over the other clustering techniques such as MDS and UMAP. PCA is well-known for its ability to minimise dimensionality while conserving underlying variation in the sample, hence preserving the genetic structure for future research (Jolliffe, 2002).

MDS, despite having equal dimension reduction capabilities, typically requires more computer resources and may not scale as well with larger datasets (Borg & Groenen, 2005). UMAP is a recent technique that provides excellent visualisation of high-dimensional data, but its interpretability in genetic clustering may be less intuitive than PCA (McInnes, et al., 2018).

The reason for adopting PCA was based on its *durability*, *interpretability*, and *computational efficiency* for big genetic datasets.

PCA Implementation Code:
To manage and analyse the genetic data, a Python script was used, which made use of many well-known libraries. The script began by importing the required modules: *Pandas* for data processing, *Scikit-allel* for genetic data handling, and *Scikit-learn* for PCA computation. The PCA results were rendered graphically using the visualisation tools *Seaborn* and *Matplotlib*, with *NumPy* supporting the mathematical computations. The *SciPy* library was used for distance computations, which are essential for understanding genetic diversity.

The method began by loading information and merging datasets to match individual samples to population labels. To guarantee a representative analysis, only 10 individuals were chosen from each population. The genetic data in VCF format was then read and transformed into a dose matrix, which served as the PCA input.

PCA was used to reduce the data to *two* major components that represented the most significant variation, and the results were recorded in a DataFrame. To increase clarity, this DataFrame was annotated with population and superpopulation labels. The PCA results were then visualised, with different plots for each 5 superpopulations. Finally, the PCA results were saved in a *TSV* (Tab Separated Values) file to ensure repeatability and accessibility in future studies. More information about this step, I will be talking about in the following subchapters.

4.1.3 Database, Visualization & Integration

The PCA data was saved in a TSV format which allowed the populating script to directly reference the PCA results to populate the database, resulting in faster data flow through the analytical pipeline. The next step was to write a backend function in the Flask application that would use the PCA data for user-driven visualisation. The Python software created for this purpose loads the PCA data, and depending on the user's options, generates a plot indicating genetic grouping.

The `process_clustering()` function collects user input from the form, determines the selection type, and calls the main plotting function. The results, which are contained in the output file name, are then passed to the results template, which allows the visual output to the user. When the user visits the *Clustering Analysis* webpage, they are presented with a selection page where they have the option of doing a superpopulation or population analysis as well as selecting a specific region for the inquiry. After submitting the analysis request, the Flask App generates and displays the appropriate PCA plot, providing an immediate graphical representation of genetic clustering based on the user's chosen choices in the user-selection page.

4.1.4 Challenges, Limitations and Future Development

Challenges Encountered:
One major issue when it came down to PCA database populating was the issue with the path themselves, as when trying to populate it would not populate to appropriate database within the repository. The issue stemmed mostly from path management in the Python script, where the code created a new *instance* folder and populated a new database rather than updating the *existing* one intended for analysis. This issue highlighted the challenge of working in a repository that requires specific route setups. To remedy this issue, changes to the script's route requirements were required, highlighting the importance of exact path management in software systems where database integrity is crucial.

The Flask application was developed in an iterative process that involved *trial and error*, particularly when connecting backend functions to the front-end. There were several errors, ranging from improper file paths resulting in missing graphs to incorrect file names causing display problems. The resolution of these difficulties was intricately linked to the repository's fundamental structure. A thorough restricting of the repository has minimised these difficulties, enhancing the Flask app's functionality and user interface interactions.

Limitations:
Although the PCA clustering algorithm was robust, it had limitations in terms of *scalability* and *adaptability*. The current solution is intended for a certain dataset structure and analytical workflow; therefore it may not be appropriate for datasets with different formats or that require alternative clustering algorithms. Furthermore, the static nature of the offered visualisations limits users' interactive analysis of PCA results to pre-generated graphical representations.

Another possible limitation encountered was that when doing the PCA analysis and plotting the graph for the Superpopulations, the function was intended to make all five plots to be plotted at the same time, however due to possible error it did not, so the generation of those five Superpopulation plots had to be generated separately/manually.

Future Developments:
Future versions of the software are intended to increase the scalability of PCA analysis, allowing for more efficient handling of large datasets. This augmentation could be performed by improving the existing codebase or utilising additional computation resources, such as *cloud computing platforms*, which have been shown to significantly improve computational efficiency and scalability (Armbrust, et al., 2010). Improving the software's versatility to accept several data types and including a variety of clustering algorithms would make the tool more versatile and relevant to a wider range of research objectives (Berkhin, 2006).

Another addition for future versions would be the addition of *interactive visualisation tools* which promise to dramatically improve the user experience by allowing for real-time interaction with PCA results. These would enable users to engage with the data in real time, resulting in a greater understanding of the underlying genetic structures (Murray, 2013). This strategy not only facilitates data exploration but also aids in the explanation of complex datasets with *intuitive graphical representations* (Heer & Shneiderman, 2012). However, when it comes to improve the user experience that usually necessitates the addition of powerful *automatic error handling* and *path resolution tools* to the software.

By doing that, this ensures that the system will proactively address common file path and database connectivity issues, hence speeding up the analytical workflow (Goodman, et al., 2014). Improving the software to intelligently manage repository topologies and automate file placement changes can also significantly reduce setup and operational challenges, resulting in a much more seamless user experience (Lourenco, et al., 2019). Finally, future developers may investigate the use of other clustering techniques than PCA, such as *t-SNE* or *UMAP*. These methods offer alternative approaches to dimensionality reduction and have been recognised for their utility in a wide range of data analysis scenarios, providing unambiguous, interpretable representations of complex data structures (var der Maaten & Hinton, 2008). This will enhance the ability for future developers to pick between these complex techniques by allowing them to select the optimal method for their specific data analysis needs, improving further the tool's usefulness (Becht, et al., 2019).

4.2 Admixture Analysis

In the context of admixture, individuals within a dataset are modelled as having portions of their genome originating from multiple ancestral populations via various genetic markers and statistical estimators. The objective here is to understand the ancestry proportions contributed by each of these source populations. These inferred proportions then can be used to produce concise visual representations that summarise an individual's genetic ancestry composition and therefore, demonstrate the existence of population structure within the data.

4.2.1 Technologies

ADMIXTURE:
The approach presented here is based on *ADMIXTURE 1.3.0* software (obtained from https://dalexander.github.io/admixture/download.html) designed for the maximum likelihood estimation of individuals' ancestries in a model-based manner from large autosomal (single nucleotide polymorphism) SNP genotype datasets, in which the individuals are unrelated. ADMIXTURE employs a block relaxation method to alternately update ancestry fraction parameters. Every update within the block is managed by addressing numerous separate convex optimization challenges through the application of a rapid sequential quadratic programming technique. The convergence pace of the algorithm is enhanced by a *quasi-Newton acceleration strategy*. This algorithm significantly surpasses the performance of both EM algorithms and MCMC sampling techniques to a considerable extent.

While ADMIXTURE also uses the same statistical model as STRUCTURE, it was chosen for its capability to calculate estimates at a faster rate using a rapid numerical optimization algorithm. Another reason admixture was picked, is because it supports parallel processing, allowing it to efficiently use multiple processors for faster analysis of large datasets.

It also offers several advanced features that enhance its versatility (Alexander, et al., 2009). For example, it can estimate the number of underlying populations (K) through cross-validation, making it easier for users to choose the appropriate model complexity. Additionally, ADMIXTURE offers a more user-friendly interface and requires less computational resources compared to some alternative options (Alexander, et al., 2009). Its ease of use and efficient resource utilisation make it particularly appealing for researchers with limited computational expertise or access to high-performance computing resources (Alexander, et al., 2009).

PLINK:
PLINK v1.90b7.2 was also employed to perform some basic manipulations of the data and provide the input for admixture in the format of *binary PLINK* (.bed), *ordinary PLINK* (.ped), or *EIGENSTRAT*(.geno) files.

Matplotlib library:
A series of visualisation packages, in particular *Matplotlib* (Version: 3.8.3), was used for visualising the output of admixture proportion inferences.

SQLite:
SQLite (v0.14.1) was selected for database management due to its lightweight nature, self-contained, serverless, and zero-configuration requirements, making it ideal for projects with limited resources. SQLite's efficiency and speed in handling queries, updates, and storage, coupled with its broad adoption and support in various programming environments, facilitated the efficient storage, retrieval, and manipulation of genetic data within the application. By utilizing SQLite, the application ensures data integrity and provides a robust framework for managing admixture datasets, contributing significantly to the overall efficiency and user experience of the platform.

4.2.2 Methodology

For the admixture analysis, a *Python script* was used to annotate the database, linking each sample with its corresponding population code using a provided annotation file. This facilitated a comprehensive examination of the genetic relationships and population structures among the sampled groups. The choice of Python scripting over other tools such as vcftools was strategic; Python's scripting capabilities offer great flexibility and precise control over data manipulation.

This level of customization, coupled with series of command-lines such as *awk* and *sed* was critical, particularly for subsequent analytical steps, as the formatting requirements of the data can present significant challenges when employing tools like PLINK.

SNPs that have high linkage disequilibrium (LD) with each other contain overlapping information, the greater concern was the possibility for certain genomic regions to exert a disproportionate impact on the outcomes, leading to a misrepresentation of genome-wide structure. To address this issue, it was decided to prune out SNPs based on pairwise LD, resulting in a minimised set of markers that are more independent from each other.

PLINK command was used to produce a new LD-pruned dataset generating two sets of files, *pruned_data.prune.in* containing a list of SNPs to keep and *pruned_output_prefix* containing a list of SNPs to exclude. This approach specifies a chromosomal window size of *50 SNPs* at a given time, and for any pair whose genotypes have an association *r2 value greater than 0.2*, it removes an SNP from the pair, in result the window is then shifted by *5 SNPs*.

```
plink --bfile "input_data”" --indep-pairwise 50 5 0.2
plink --bfile "input_data" --extract "pruned_data.prune.in" --make-bed
 --out "pruned_output_data"
```

Figure 3. *PLINK Command-Line Instructions for Data Pruning.*

The parameter K in population genetics refers to the number of assumed ancestral populations used to model the genetic structure of the data. It plays a critical role in genetic ancestry analysis as it determines the level of granularity in identifying population substructure. Selecting the appropriate K value is crucial for accurate inference of population structure and can have significant implications for the interpretation of genetic data.

A *smaller K* may represent broad population categories, while a *larger K* can reveal finer-scale subpopulations. The choice of K involves a *balance* between model complexity and fit to the data, as higher values of K may risk overfitting. The lowest cross-validation error, often achieved at the optimal K value, indicates the most reliable model fit.

ADMIXTURE's cross-validation procedure was used to choose an optimal K value by comparing cross-validation errors for different K values. To do so, the following command line was employed to run the ADMIXTURE for values of K ranging from 1 to 8 using 5 threads for parallel processing.

```
for K in {1..10}; do
    admixture --cv "pruned_output_data" $K -j5 | tee
    "Admixture_result/log${K}.out"
done
```

Figure 4*. Script for Executing ADMIXTURE with Cross-Validation to Determine Optimal K Value.*

The cross-validation error was assessed for each K value and the CV error for *K=3* was found to be the lowest, with a value of *0.10485* (as shown in figure 5). This suggests that *K=3* provides the best fit for the dataset among the tested range of K values. Additionally, figure 5 shows how the CV error changes as K increases, revealing the pattern of elbow points.

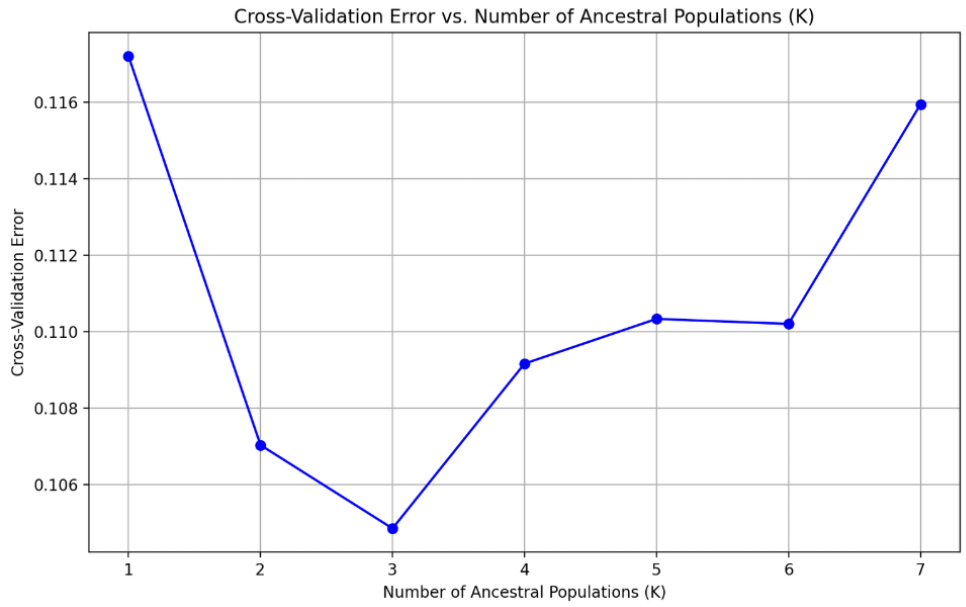

***Figure 5**. Cross-Validation Error by Number of Ancestral Populations. This graph illustrates the relationship between the number of ancestral populations assumed (K) and the cross-validation error obtained from the ADMIXTURE analysis. The error is minimized at K=3, suggesting an optimal fit at this number of populations, whereas higher K values result in increased error, indicating a potential overfitting of the model to the data.*

The K value directly influences the interpretation of population structure in genetic data. By aligning K with the known biological context—in this case, the presence of *five super populations*—it is ensured that the admixture analysis reflects the true genetic diversity and differentiation among these populations. Therefore, in addition to K=3, *K=5* was also selected to match the number of superpopulations and adhere to biological relevance.

ADMIXTURE was run with both K=3 and K=5 producing *two major output files* with *suffix .P* containing the allele frequencies inferred for each SNP in each population. The file with *suffix .Q* contains the inferred individual ancestry proportions from the K ancestral populations, with each individual represented by a single row.

```
admixture "pruned_output_data" 3
admixture "pruned_output_data" 5
```

***Figure 6**. ADMIXTURE Commands for Analysis with K=3 and K=5.*

4.2.3 Database and Visualization

Upon completing the admixture analysis, the ancestry proportions for each individual, derived from the *.Q* output files, were annotated with sample ID in the same order using *python_to_merge_q_code* and *sample_ID.py* and systematically loaded into the schema database using *admixture_populating.py*. This integration was executed using a Python script, which was carefully designed to match the format and structure outlined in section 3, *Database Schema*. The script ensured that the data was accurately parsed and inserted into the database, allowing for efficient retrieval and manipulation for downstream analysis.

Furthermore, the successful loading of this data into the database facilitated the subsequent visual representation of the ancestry proportions. The visualization was achieved through a combination of Python's matplotlib library and custom scripting. Initially, two sets of Python scripts were employed to categorise individuals according to their population and superpopulation groupings, effectively organising the data for visualisation. This was achieved through a combination of Python's matplotlib library and custom scripting. This preparatory step was crucial for the generation of two distinct sets of graphs, one representing the results for K=3 and the other for K =5, as shown in Figures 7 and 8, respectively.

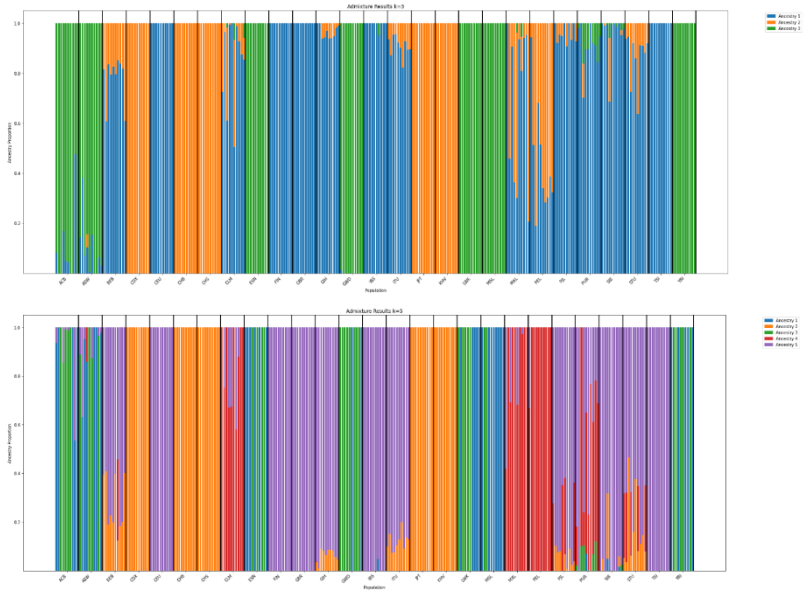

Figure 7*. Admixture Proportion Distributions by Population shows the genetic ancestry proportion for each individual, grouped according to their respective population.*

BECAUSE OF FORMATTING ISUES, FIGURE 8 WILL BE SHOWN IN THE NEXT PAGE.

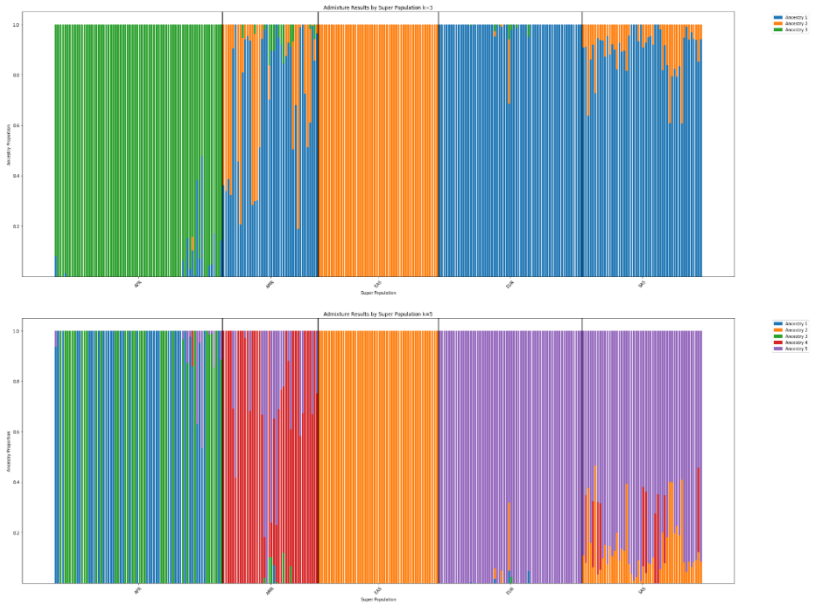

***Figure 8**. Admixture Proportion Distributions by Population shows the genetic ancestry proportion for each individual, grouped according to their respective superpopulation.*

4.2.4 Admixture Analysis Integration

The initial implementation of the scripts, designed to present admixture results, displayed the entire dataset in a single comprehensive view. While this approach provided a macroscopic perspective of the genetic data, it became apparent that such an extensive display could lead to a cluttered and overwhelming user experience. The subtle distinctions between individual populations and superpopulations were not easily distinguishable. To address this, the script was refined to *flask_admixture copy*.py to allow a more curated presentation of the data. Users of the application can now view admixture results for specific populations or super populations based on their selections and by allowing them to filter the view according to their interests, the application facilitates a more engaging and informative exploration of genetic ancestry proportions.

A key integration point of the module within the application is the retrieval and visualising of the admixture result based on user input. Error handling within the module is designed to be user-proof, given that input options are pre-determined and not susceptible to invalid entries. This approach helps eliminate the possibility of invalid inputs, spelling errors and, ensuring a smooth user experience.

Data flow within the application is designed to be both fast and efficient. Upon the user's selection of a specific population or super population, the *app.py* script retrieves the pre-estimated admixture results from the database. This process is mediated through several Python functions and SQLite, which ensure the data is processed and visualised. The resulting visual output, generated via the matplotlib library, is then displayed to the user, offering a representation of the ancestry proportions of each individual within the dataset.

Performance optimization and accuracy were achieved by precomputing and storing the admixture results in the database. This approach ensures that the application remains responsive to user queries, as it eliminates the need for real-time data processing which could introduce latency.

Furthermore, this strategy minimises the potential for computational errors during user interactions, as the complex and resource-intensive computations are overseen as a priority, and users are provided with immediate access to pre-validated results.

4.2.5 Challenges, Limitations and Future Development

Managing Large Datasets and Computational Constraints:
To navigate the limitations imposed by computational resources and time constraints, a strategic subset of 270 individuals was selected from a larger pool of 4,000. This ensured a random yet representative sampling from each of the 27 populations. Initially, selecting one individual per population resulted in a sample size of 27, which proved insufficient in capturing the genetic diversity while considering the biological contexts. Attempts to increase the sample size to 100 individuals per population highlighted the computational limitations of personal computing resources onset. Consequently, settling on 10 individuals per population struck a balance, allowing for manageable analysis. Custom Python scripts played a crucial role in data preparation, especially in ensuring compatibility with the BLINK tool, despite the challenges imposed by the required formatting. Moreover, discrepancies among the sample IDs presented an additional challenge, complicating the annotation process which was resolved with *cleaning.tsv_file.py*. Through careful manipulation and careful use of typographical symbols, each sample was annotated with its respective population code, remaining compatible with formatting requirements.

Algorithmic Efficiency in Admixture Analysis:
Efficiency in running ADMIXTURE and PLINK, particularly for multiple K values, was crucial. The computationally intensive nature of these tools, especially during cross-validation for optimal K selection, required careful planning. Although automation of the cross-validation process was straightforward, the time-consuming nature of this task limited the ability to perform concurrent analyses. Future analyses could benefit from leveraging high-performance computing clusters to ease these time constraints and improve efficiency.

Visualisation and Integration Challenges:
Designing an effective visualisation scheme was paramount, given the complexity of conveying genetic ancestry proportions. The development of iterative loops for compiling individual ancestry data required careful trial-and-error efforts to ensure completeness and accuracy in the graphical representations. Integrating these visualisations into the Flask-based application posed additional challenges, particularly in managing file paths and directories for an effective data flow. Debugging and close examination were essential to resolve often cryptic error messages and ensure a smooth user experience.

Future Directions and Development:

Increasing the scope of analysis to include larger datasets and a more diverse array of populations could significantly enrich our understanding of genetic diversity and admixture patterns. The adoption of advanced computational methodologies and the integration of cloud-based infrastructures are proposed to address existing constraints, paving the way for more profound analytical approaches. Ensuring a "user-proof" design for input forms and error handling has been paramount for a smooth user experience in this software. Looking forward, not only is there potential to add more populations to the dataset, but we could also explore enabling users to input their own populations or super populations of interest. This enhancement would require error handling and validation processes to maintain data integrity and user experience.

Another avenue to be considered, is offering users the ability to select from a range of analytical tools for admixture analysis, such as *PopCluster* and *Neural ADMIXTURE*. PopCluster, with its two-step approach combining clustering analysis and an admixture model, has shown promise in accurately inferring population structure and individual admixture proportions from genotype data across a wide range of dataset sizes (Wang, 2022). Its use of simulated annealing and an EM algorithm helps to navigate the challenges posed by small sample sizes or prominent levels of admixture, making it a versatile option for complex genetic analyses. On the other hand, Neural ADMIXTURE leverages the power of neural networks to enhance the speed and precision of clustering predictions.

This method stands out for its ability to process extremely large datasets in a fraction of the time required by traditional methods, while also providing clearer differentiation between closely related populations (Mantes, et al., 2023). Incorporating a feature on the website that allows users to choose between such advanced tools based on their specific project needs or computational constraints could significantly enrich the user experience.

4.3 Retrieval of SNP Genetic Information

The application is designed to allow users to retrieve genetic information that seamlessly integrates with the broad objectives of Arch Genome to analyse population genetic structure and investigate relationships with diverse human populations. This functionality provides users, especially clinicians and genetic researchers, accessibility to detailed insights into the genetic composition of populations, specific genomic regions, and identification of potential clinical implications. Users have the flexibility to select the populations of interest for retrieval of genetic information, ensuring their analysis is tailored to specific demographic or geographic groups. This adaptability enhances the precision and relevance of the obtained genetic insights. Key features encompass SNP-specific information, enabling users to input unique SNP identifiers to explore corresponding allele and genotype frequencies, alongside clinical relevance if available. Additionally, it allows users to define the genomic region of interest (based on Chromosome 1) by defining the start and end coordinates. Also, the user can extract genetic information by providing specific gene names, facilitating the identification of potential associations with both population structures and clinical relevance.

4.3.1 Technologies

ANNOVAR (ANNotate VARiation):
ANNOVAR software is a specialised bioinformatics tool that facilitates quick and easy variant annotations, including gene-based, region-based, and filter-based annotations based on VCF file input (Wang, et al., 2010). This tool was used to obtain the gene names required for user selection input for retrieving SNP information and Population Differentiation Matrix. Additionally, the clinical relevance annotations were obtained and represented by 3 key components: *CLNALLELEID* (clinical allele ID), *CLNDN* (clinical disease name), and *CLNSG* (clinical significance).

Besides ANNOVAR, the *SNP effect* (SnpEFF) and *Variant Effect Predictor* (VEP) are commonly used annotation tools. In comparison, ANNOVAR offers the easiest installation as it unpacks the downloaded file and provides rich documentation and extensive flexibility for users to retrieve a variety of annotation types according to their own needs. The selection of ANNOVAR is widely supported by clinical genome-sequencing studies, as it was featured in the *2014 CLARITY report* (an international effort toward developing standards for best practices in analysis, interpretation, and reporting of clinical genome sequencing results). Overall, there was a total of **52%** participating clinical genome sequencing laboratories and **63%** of finalists that used ANNOVAR to discover disease-related mutations (Brownstein, et al., 2014).

Although, the limitation of this software is the lack of a common standard for denoting functional annotations. This is because ANNOVAR adopts its own nomenclature towards functional annotations, as well as its specific output file formats. Thus, it may cause difficulty to draw result comparisons with other annotation software tools, or implementation by further downstream analysis tools (Yang & Wang, 2015).

VCFtools:
VCFtools is a versatile program package that performs genetic data analysis applied to VCF files. This had a crucial role to determine the allele frequency of each population, which is required for user selection input for retrieving SNP information and serving as a fundamental component for calculating the FST value in Population Differentiation Analysis. Despite VCFtools having an established function to compute the FST statistics between population pairs, this feature was not implemented in our application.

The decision was based on the complexity arising from the magnitude of population combinations, which suggests impracticality for the database to store all those results. Instead, an alternative approach was adopted as discussed in section 4.4. Moreover, VCFtools has a function based on computing the *Hardy-Weinberg equilibrium* (HWE) statistics, which allows determination of the genotype counts. Subsequently, these counts were converted into genotype frequencies based on the counts relative to the total number of samples at each SNP.

4.3.2 Methodology

ANNOVAR:
ANNOVAR provides quick installation of a package that can be loaded onto VScode. This includes a function to convert the original VCF dataset (*chr1.vcf.gz*) obtained from our collaborator to its new *AVINPUT* format (chr1.avinput) to be compatible with its annotation tools.

For gene-based annotations, there are many reference datasets accessible to download on ANNOVAR's website. This application is based on the most recent *hg38refGene dataset*, which was last updated on 17th August 2020 by University of California, Santa Cruz (UCSC). This is used to specify the genome build version as *hg38* that matches our original dataset, as well as proving the gene names aligned with specific genomic coordinates. This information was aligned with our dataset, which forms the key components for user selection criteria.

For clinical relevance annotations, our application is based on the *hg38clinvar dataset*, which was last updated on December 31st, 2022, by ClinVar. This provides multiple attributes aligned with genomic coordinates, such as *CLNALLELEID* (clinical allele ID), *CLNDN* (clinical disease name), *CLNDISDB* (clinical database source for disease name), *CLNREVSTAT*(review status), and *CLNSIG* (clinical significance). Upon review, the final decision was to only reserve *CLNALLELEID*, *CLNDN*, and *CLNSIG*. This justification is valid because it offers more known information compared to other entries that contain empty fields. The inclusion of *CLNALLELEID* in the dataset provides multiple advantages for the users, as it serves as a valuable reference point for each variant and allows users to easily access more detailed or up-to-date information on the original ClinVar website. This approach ensures dataset size and clarity, since *CLINALLELEID* does not significantly impact the dataset size, while keeping the primary focus on the key *CLNSIG*. Overall, this approach contributes to maintaining data integrity to avoid the necessity of frequently updating the dataset, which can be resource intensive. Likewise, the users have more flexibility and convenience to access the data source to accommodate their varying needs for clinical information of interest.

VCFtools:
For the preliminary steps, *bcftools* was used to filter the original chr1.vcf.gz file to only preserve *SNPs*. This was saved into a new file called *SNP.vcf*. To maintain uniformity with the other analyses, the sample size comprised **10 individuals randomly chosen from each population**, extracted from the original *sample_pop.tsv* file provided by the collaborator. A python script was employed to store each population sample in different text files.

These text files were then implemented into another python script that iterates each population with the employed VCFtools function. For the extraction of allele frequencies, this was applied to the -- *freq* function.

However, the extraction of genotype frequencies consists of two steps. The first step involves using the --*hardy* function to perform the *Hardy-Weinberg equilibrium* (HWE) tests. Once finished, this outputs corresponding HWE files that were subsequently input into a separate python script to extract the observed genotype counts values and calculate the genotype frequencies.

4.3.3 Challenges, Limitations and Future Development

The main challenges faced was the computational efficiency time to process the scripts. The VCFtools provides an output file to keep record of the run time for both --*freq* and --*hardy* functions. The average time to compute for each population was approximately **30 minutes**. Therefore, taking account the effort to parallelize the scripts, the overall runtime was approximately **27 hours**. Another limitation was the sample selection criteria, as it affects the completeness of data in the allele frequency and genotype values. This was due to time constraints, but future development with increasing sample size could enhance the robustness and representativeness, leading to more reliable results. Another solution to address these challenges would be to perform certain data imputation to provide an estimate of missing values. This would provide a more complete dataset and reduce the impact of sample selection bias.

4.4 Pairwise Population Genetic Differentiation

Population differentiation analysis offers critical insights into the evolutionary processes that impact genetic variation within and among populations. *Fixation index* (Fst) statistics are applied to quantify levels of genetic differentiation between populations based on the allele frequencies. There are a variety of conceptually defined methods to obtain the FST value, as originally established by *Wright's Fst estimator* as the ratio of the observed variance of allele frequencies between subpopulations to the expected variance of allele frequencies, assuming panmixia (Wright, 1965). This was later improved to new frameworks, such as Weir and Cockerham (1984) where it is widely used due to its ability to describe the genetic population structure in a single summary statistic, asymptotically unbiased with respect to sample size, and compensates for overestimates at low levels of genetic differentiation, unlike Wright's estimator.

4.4.1 Technologies

SQLite Database:
The database plays a pivotal role in data storage and retrieval, containing required information such as allele frequency and SNP information. This serves as a robust foundation to facilitate efficient data querying for subsequent downstream analyses.

Python:
This is used for scripting and automation throughout the software pipeline. This is crucial for managing tasks such as *data retrieval, allele frequency, FST calculations,* and *data visualisation*. The script incorporates python libraries, such as *sqlite3, pandas, seaborn*, and *matplotlib*. This ensures streamlined data manipulation, statistical analysis, and creation of visually informative population differentiation matrices.

File Handling:
The *os* module is employed for file and directory manipulation. The *sanitize_filename* function ensures the filenames are properly formatted.

4.4.2 Methodology

Database interaction:
In the database handling phase, genetic data undergoes storage and retrieval processes within an SQLite database. The sqlite3 library is used to interact with the SQLite database. For instance, functions like *get_allele_frequency* retrieves the allele frequency for the given user SNP ID input and population codes. The *calculate_fst* function calculates the pairwise FST values for each pair of populations based on allele frequencies.

Fst Calculations:
Python scripts systematically iterate through specified genes and populations, extracting genetic data and employing mathematical computations to determine allele frequencies. Subsequently, Weir and Cockerham's FST is calculated to assess genetic differentiation between populations, executed by Python scripts conducting pairwise FST calculations. The results are stored in the *fst_values*.

FST Results CSV File:
The calculated FST values are organised into a panda DataFrame, and the data frame is saved to a CSV file using the *to_csv method*. This CSV file contains information about *SNP IDs, population pairs,* and their *corresponding FST values*. The CSV file serves as a structured and portable format for storing and sharing FST results.

Heatmap Image Files:
The FST values are visualised as heatmaps using seaborn and matplotlib, and the resulting heatmaps are saved as image files (PNG format). The *create_fst_heatmap* function organises the FST DataFrame and generates a heatmap image file for each SNP ID. These image files are stored in the *static/heatmap* directory.

4.4.3 Challenges, Limitations and Future Development

When assessing the performance of the website, the average time to load the population differentiation based on three populations was approximately **2 minutes**. However, this is dependent on the number of populations selected, as a greater selection of populations would prolong the runtime. To address this issue, the python library *multiprocessing* could be implemented to facilitate parallelisation of the population differentiation calculations. Additionally, if the user requests differentiation for a set of populations that has been processed before, these results can be retrieved quicker by using a *cache system* to avoid redundant calculations. Furthermore, the application could incorporate a progress indicator to enhance user experience by informing about the ongoing process and estimated completion time.

5. Conclusion

Team Arch's AG-ArchGenome project is a significant step forward in population genetics, providing a robust web platform for in-depth genetic data analysis. This platform, which includes innovative analytical approaches such as *Principal Component Analysis* (PCA), *Admixture Analysis*, *SNP Information* and *Population Genetic Differentiation*, is able to give academics and geneticists a powerful tool for deciphering the complexities of population structure.

Despite substantial challenges due to large datasets and computational constraints, the study effectively solved these concerns by selecting a representative subset of individuals, ensuring both computational manageability and genetic diversity preservation. The application's visualisations and seamless integration with a Flask-based web platform have improved the user experience, allowing for intuitive and in-depth genetic data analysis. Furthermore, the project's adaptability in methodological execution and scalability in its software framework are evident in its successful navigation of data path management challenges and iterative development of the Flask application.

The AG-ArchGenome project shows the successful collaboration between computational biology and software development. It met its objectives by providing a comprehensive, efficient, and user-friendly platform for genetic data analysis. The project's capacity to overcome computational challenges, incorporate complex analytical tools, and deliver a seamless user experience demonstrates its overall success.

Looking ahead, the project is poised for further improvements, such as the incorporation of more advanced computational methodologies, the expansion of the dataset to include a wider range of populations, and the introduction of additional analytical tools to meet a broader range of research needs. The current version of AG-ArchGenome provides a solid foundation for future development, with even greater contributions to the field of population genetics.

References

Alexander, D. H., Novembre, J. & Lange, K., 2009. Fast model-based estimation of ancestry in unrelated individuals. *Genome Research,* 19(9), pp. 1655-1664.

Alexander, T. A. & Machiela, M. J., 2020. LDpop: an interactive online tool to calculate and visualise geographic LD patterns. *BMC Bioinformatics,* 21(1), pp. 1-4.

Armbrust, M. et al., 2010. Clearing the clouds away from the true potential and obstacles posed by this computing ability. *Communications of the ACM,* 53(4), pp. 50-58.

Becht, E. et al., 2019. Dimensionality reduction for visualising single-cell data using UMAP. *Nature Biotechnology,* 37(1), pp. 38-44.

Berkhin, P., 2006. In Grouping multidimensional data. In: N. C. T. M. Kogan J., ed. *A Survey of Clustering Data Mining Techniques.* s.l.:Springer, pp. 25-71.

Borg, I. & Groenen, P. F., 2005. *Modern Multidiemnsional Scaling: Theory and Applications.* Berlin: Spring Science & Business.

Brownstein, C. A. et al., 2014. An international effort towards developing standards for best-practices in analysis, interpretation and reporting of clinical genome sequencing results in the CLARITY Challenge. *Genome Biology,* Volume 15.

Dutheil, J. Y., 2021. Correction to: Statistical Population Genomics. *Methods in molecular biology,* Volume 2090, pp. C1-C1.

Goodman, A. et al., 2014. Ten SImply Rules for the Care and Feeding of Scientific Data. *PLOS Computational Biology,* 10(4), p. e1003542.

Heer, J. & Shneiderman, B., 2012. Interactive Dynamics for Visual Analysis. *Communications of the ACM,* 55(4), pp. 45-54.

Jolliffe, I., 2002. *Principal Component Analysis.* 2nd ed. s.l.:Spring Series in Statistics.

Kitada, S., Nakamichi, R. & Kishino, H., 2021. Understanding population structure in an evolutionary context: population-specific FST and pairwise FST. *G3 Genes|Genomes|Genetics,* 11(11), p. jkab316.

Liu, C.-C., Shringarpure, S., Lange, K. & Novembre, J., 2020. Exploring Population Structure with Admixture Models and Principal Component Analysis. *Methods in molecular biology,* Volume 2090, pp. 67-68.

Lourenco, J. R. et al., 2019. Choosing the right NoSQL database for the job: a quality attribute evaluation. *Journal of Big Data,* 6(1), pp. 1-46.

Mantes, A. D. et al., 2023. Neural ADMIXTURE for rapid genomic clustering. *Nature Computational Science,* 3(7), pp. 621-629.

McInnes, L., Healy, J., Saul, N. & Grobβberger, L., 2018. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software,* 3(29), p. 861.

Murray, S., 2013. *Interactive Data Visualization for the Web.* 2nd ed. s.l.:O'Reilly Media, Inc..

Rosenberg, N. A. et al., 2005. Clines, Clusters, and the Effect of Study Design on the Inference of Human Population Structure. *PLOS Genetics,* 1(6), pp. e70-e70.

Rosenberg, N. A. et al., 2002. Genetic structure of human populations. *Science,* 298(5602), pp. 2381-2385.

var der Maaten, L. & Hinton, G., 2008. Visualising Data using t-SNE. *Journal of Machine Learning Research,* Volume 9, pp. 2579-2605.

Wang, J., 2022. Fast and accurate population admixture inference from genotype data from a few microsatellites to millions of SNPs. *Heredity,* 129(2), pp. 79-92.

Wang, K., Li, M. & Hakonarson, H., 2010. ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic Acids Research,* 36(16), p. e164.

Weir, B. S. & Cockerham, C. C., n.d. Estimating F-Statistics for the Analysis of Population Structure. *Evolution.*

Wright, S., 1965. The Interpretation of Population Structure by F-Statistics with Special Re. *Evolution,* 19(3), pp. 395-420.

Yang, H. & Wang, K., 2015. Genomic variant annotation and prioritization with ANNOVAR and wANNOVAR. *Nature Protocols,* Volume 10, pp. 1556-1566.

Zhou, G. et al., 2022. MSXFGP: combining improved sparrow search algorithm with XGBoost for enhanced genomic prediction. *Heredity,* 129(2), pp. 79-92.