NixOS Software Documentation

A documentation submitted as part of the requirement for the
MSc Bioinformatics Software Development Group Project

NixOS Team
Ellie Gadson, Ceri Harman, Cian Kennedy, Natalia Zemla

Contents

1.0 Introduction

1.1 Purpose

The Software Documentation describes the architecture and system design for NixOS, a molecular biology web-based software. NixOS is designed to handle population genetics data, by retrieving information from a database (mySQL) and running structure analysis on populations and superpopulation via clustering analysis (PCA), admixture analysis (ADMIXTURE) and pairwise genetic differentiation (Fst). This documentation is intended for Software Developers and anyone else interested in reproducing and testing this software.

1.2 Scope

This document describes the implementation details of a NixOS web-based software. NixOS consists of six major components: Framework, Database, Clustering Analysis (PCA), Admixture analysis (ADMIXTURE), Allele and Frequencies and Pairwise population genetic differentiation analysis (Fst). Each of these components will be explained in detail in this Software Documentation.

1.3 Definitions

Term	Meaning
PCA	Principal Component Analysis
Fst	The Fixation Index, F-Statistics
PVE	Proportion of Variance Explained
HPC	High Performance Computing

2.0 System Overview

2.1 Software Architecture

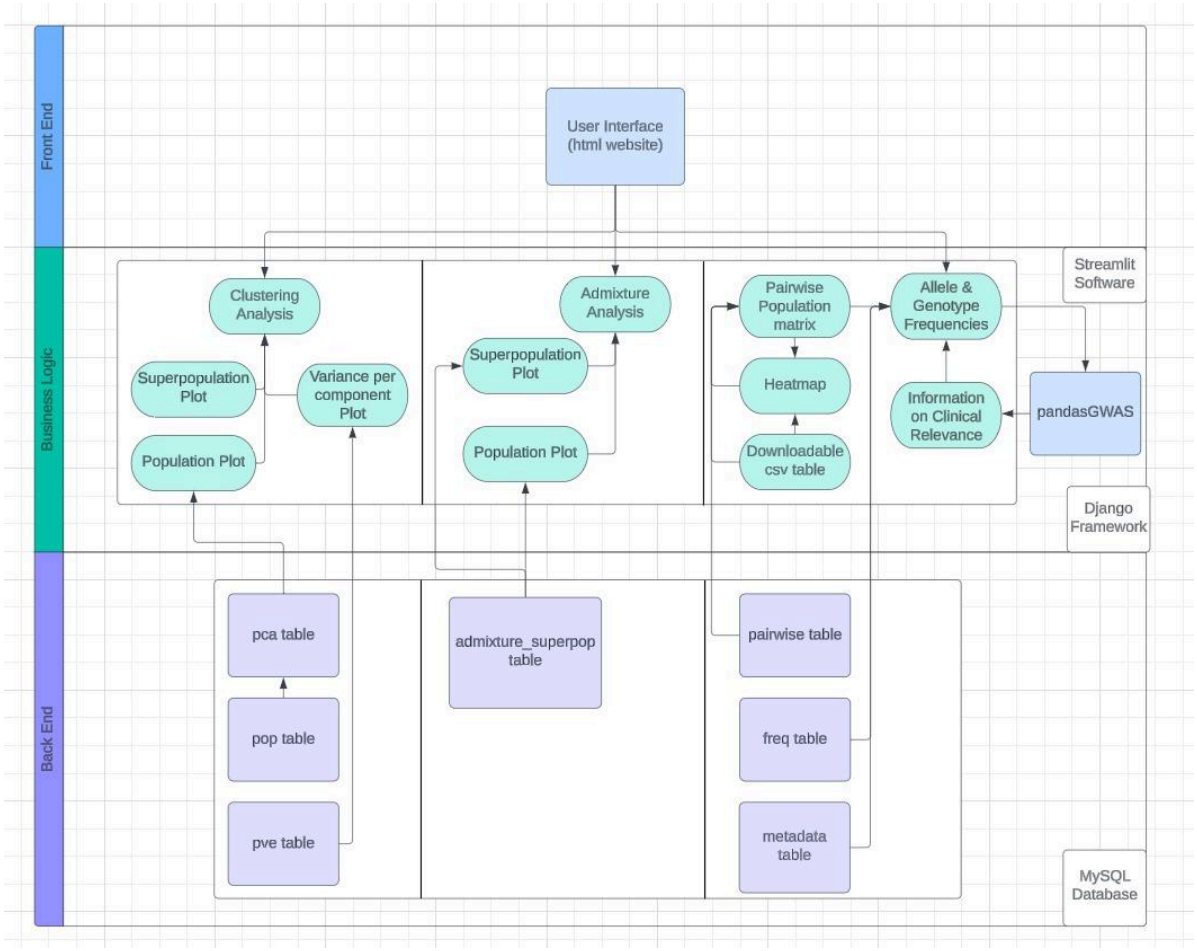

Figure 1. NixOS software architecture structure

The structure of the software was designed in three sections; back end, business logic and front end. The ‘backend’, as seen in *figure 1*, resided in our SQL database. This contained the data required to run the analysis components of the NixOS software, as well as the metadata for chromosome 1 e.g. population names, gene names, coordinates and SNP IDs. This ‘backend’ information is called upon in our ‘business logic’, when requested by the user. The ‘business logic’ section is our Django framework, streamlit dashboard and pandaGWAS. This is quintessential to our software as it creates the actual application, while linking it to pandasGWAS and the MySQL database; pandasGWAS being used for clinical relevance on the allele and frequencies page. The ability of Django and streamlit to produce dashboards, made the ‘user interface’ integration of our application seamless, this in addition to the ‘html website’ created our ‘front end’ section.

3.0 System Components

3.1 Database

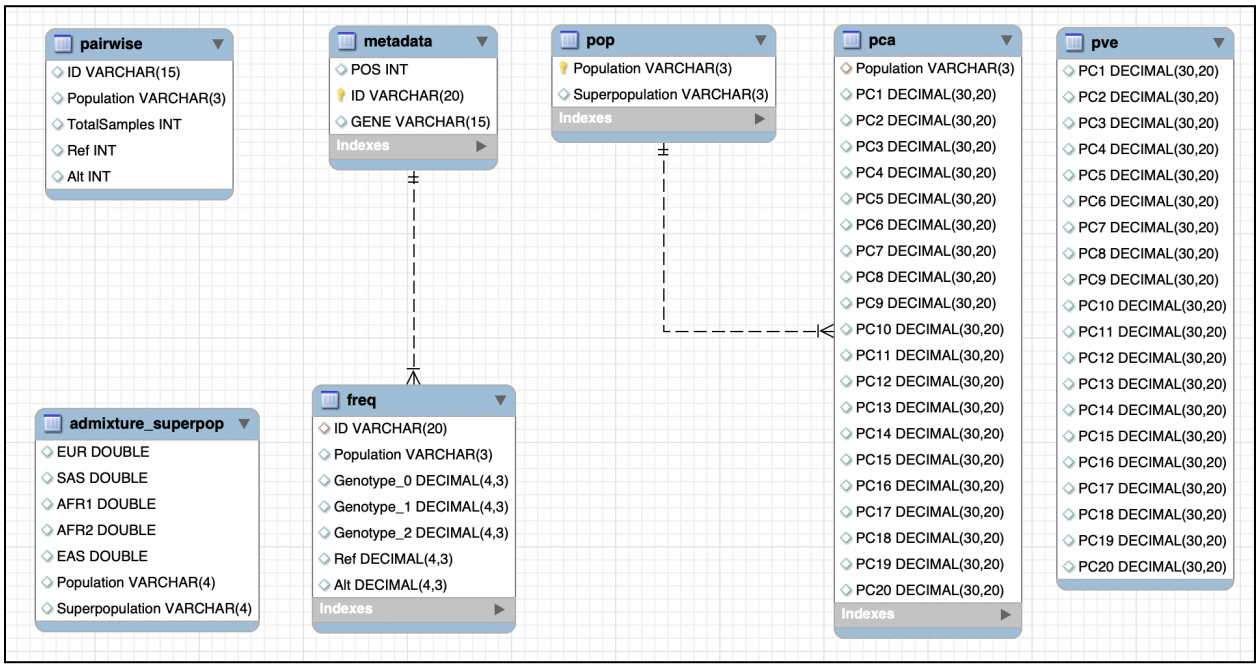

Figure 2. SQL Database SCHEMA visualised in MySQL Workbench 8.0

A relational database management system was used for storing data for the analyses. The database was built in MySQL for ease of use and integration with development tools, as well as the potential for multi-user access and scalability. The database stores information that is queried for each of the analyses. It stores results of PCA and admixture analysis, PCA proportion of variance explained (PVE), genotype and allele frequencies as well as allele counts used for pairwise differentiation analysis (see *figure 2*). Most tables contain an identifying variable such as SNP ID or population. Additionally, the database stores information about SNP positions and gene names.

The “metadata” table stores data about the position of a SNP on the chromosome (“POS”), SNP ID (“ID”) and gene name (“GENE”). This information is used in the user selection process of genotype and allele frequencies as well as pairwise differentiation analysis. The “freq” table has a foreign key relationship with the “metadata” table that references its “ID” column. The frequencies of the genotypes and alleles (rounded to 3 decimal places), as well as SNP ID and population data, are all contained in the “freq” table. This design allows for efficient querying of the “metadata” table for either gene names or SNP positions, which can be used to select the desired SNP IDs and the frequency values from the “freq” table. It also maximises storage efficiency thanks to a lack of position and gene name repeats, which would be the case if they were contained in the “freq” table.

The “pop” table contains existing populations and their associated superpopulations, stored as 3-letter codes (e.g. GBR for Great Britain). Its purpose is to store superpopulation data so that only population data is needed in other tables. Currently the “Population” column of the “pop” table is referenced with a foreign key by the “pca” table, so that when a superpopulation is queried, the necessary populations of the “pca” table are selected. The “pop” table can be easily integrated with the “freq” (for additional options for the user) and “admixture_superpop” (for improved storage efficiency) tables, however, due to time limitations of the project and other factors, this has not been achieved.

The “pca” table contains PCA values from PC1 to PC20 (rounded up to 20 decimal places), as well as population data. This design is optimal for minimising redundancy of data (column for each PC instead of PCs being stored in rows), precision and querying efficiency.
The “pve” table stores PCA proportion of variance explained data. For easy integration with the interface, it contains 20 columns, from PC1 to PC20, similarly to the “pca” table.
In the “pairwise” table, reference and allele counts of each SNP from each population are contained, alongside a sum of the reference and allele counts. These values are used for pairwise genetic differentiation analysis.

The “admixture_superpop” table stores results of the admixture analysis. Five of the columns represent the superpopulations the results resemble for ease of interpretation and integration. The two other columns are “population” and “superpopulation”, which contain data about the region the admixture analysis results belong to. This data is used for visualising admixture analysis. To reduce storage necessary for the database, the “superpopulation” column could be replaced by a foreign key to the “pop” table; however, as the aim of the design was simplicity and efficiency of the web app, this has not been implemented.
All in all, the design of the database has met the set requirements. It was optimised to contain as little redundant and unused data as possible, which minimises necessary storage. It also allows for straightforward querying of the data which greatly improves the ease of the integration with other softwares as well as the web app’s efficiency.

3.2 Interface Description

The Interface was designed with useability heuristics in mind to create a readily accessible platform for the user. Whilst Django does not require a frontend software, we integrated Streamlit due to the various built-in resources which improve the user experience (for example, user- friendly input options and an automatic loading icon to keep the user notified of the app’s status). The design focussed on minimalism, error prevention (over error messages) and consistency. These topics will be discussed specifically with examples in the following sections, but a brief overview is provided here:

- Minimalism, reducing clutter and the amount of non-vital information displayed to the user aids in their experience and their interpretation of the data.
- Error prevention, while Molich & Nielsen, (1990) discusses the importance of clear error messages they highlight the precedence of prevention via careful design.

- Consistency, the use of the clear and uniform language and tools should reduce burdens on the user and improve the journey.

3.3 Home Interface

The purpose of this page is to carry the bulk of the written information the user needs to understand the context of the analyses and the app. This was achieved by laying out the information as clearly as possible by spacing out the text, highlighting key words and keeping a consistent layout.

However, there is a delicate balance between cognitive load and useability which may be improved upon in the following dashboards by creating a shortcut whenever a superpopulation is mentioned to show the user what populations are included. Thus, giving all the information to the user when needed without the user having to return to the homepage.

3.4 Clustering Analysis Interface

Figure 3. Graphical user interface for the clustering analysis page, PVE plot and user component selection.

The Clustering Analysis dashboard was designed with simplicity and ease of use in mind. The design object of this page was to balance cognitive load and usability, in particular this was achieved by reducing the amount of text and displaying only the needed information which is

clearly demonstrated by the graph “Variance captured per component” (as shown in *figure 3*). This graph instantly communicates the data compared to a table presenting numerical values. If the user wishes to obtain specific % of variance, this can be done by interacting with the graph.

Streamlit’s built-in “expander” function allows for entire sections to be hidden until required. This reduces clutter on the dashboard by not displaying graphs and options needlessly. This “expander” function was utilised throughout the dashboards of this app for the same effect.

The Population or Superpopulation analysis options, when expanded, allow for the user to select which component may be selected for each axis. By providing a series of components to select from this prevents errors which may occur if another method of selection, such as text box, was implemented.

While most of the variance can be captured within the first three components, there is a benefit in allowing the user to select which components they wish to plot. This flexibility allows users to explore the data for their own understanding and may prove useful for comparative analysis (if the similar patterns are observed across several different component selections).

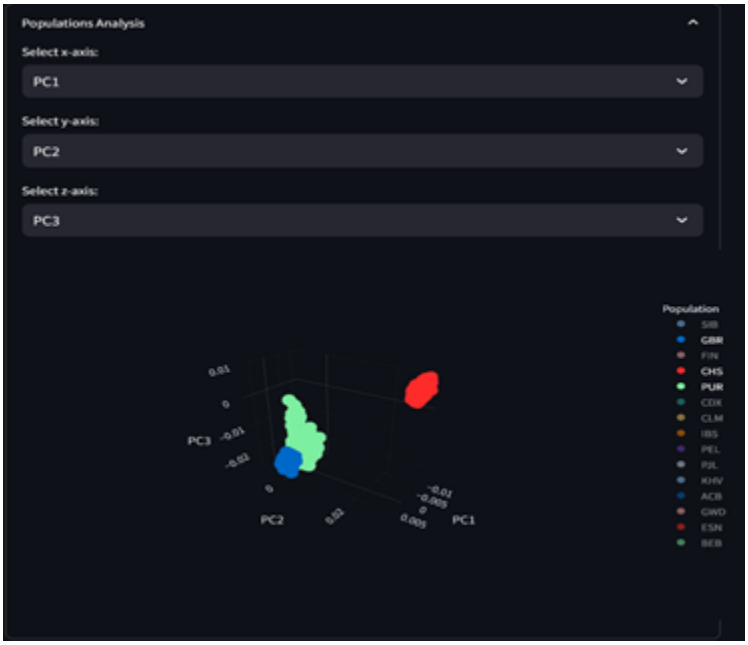

Figure 4. User graphical interface for the clustering analysis, user component selection and pca plot.

Once the graph is produced (*figure 4*), a helpful legend is included which maps each population or superpopulation to a colour. This graph is made interactive by allowing the user to manipulate the 3D image as well as selecting which populations/superpopulation they wish to include by interacting with the legend. These built-in Streamlit features aid the user experience through simplistic design and by aiding inherent understanding.

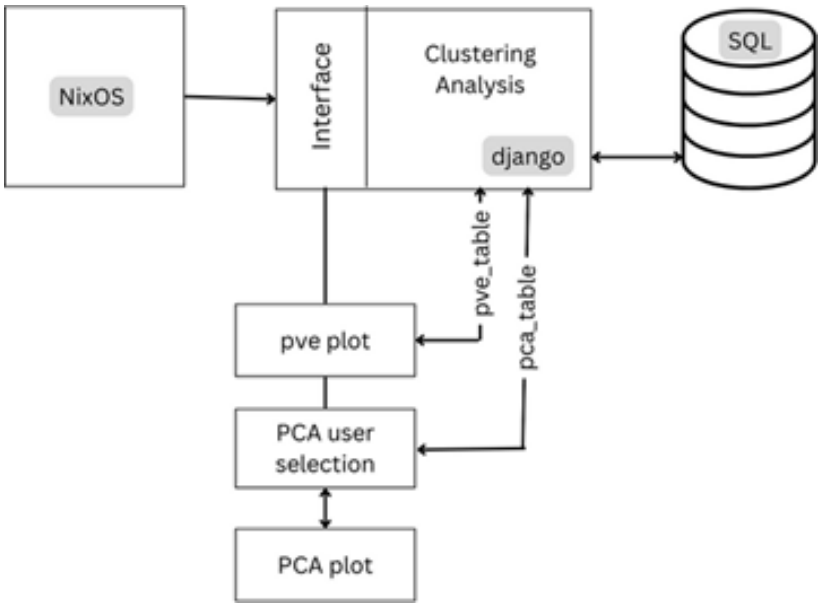

Figure 5. User interface for the clustering analysis page, interaction of application and backend.

The frontend code for this dashboard is accurately displayed in *figure 5*. Several tables are pulled from the MySQL database and stored as Pandas dataframes in the frontend. While this method can be incompatible with large datasets, as the “pve” and “pca” tables which store the data for “Variance captured per component” plot in *figure 5* and “Populations Analysis” respectively, are relatively small this method was initially found to be acceptable. If these tables were expected to expand in the future, scaling issues may arise affecting load times (another method of coding could be employed which will be discussed later in this section).

However, due to a miscommunication within the team the “Superpopulation Analysis” was the last feature to be added to the app. Once coding began, we realised that a MySQL join query between the “pca” and “pop” table proved too difficult to implement at that moment in time.

The alternative method of coding which may prove more efficient employs the use of MySQL queries to the database once the user has selected which components and populations they wish to use. This method reduces the amount of data being pulled from the database by querying it specifically. This method could reduce load times and handle larger datasets. The “pve” table might be omitted from this method, as it should in theory always stay quite small.

3.5 Admixture Analysis Interface

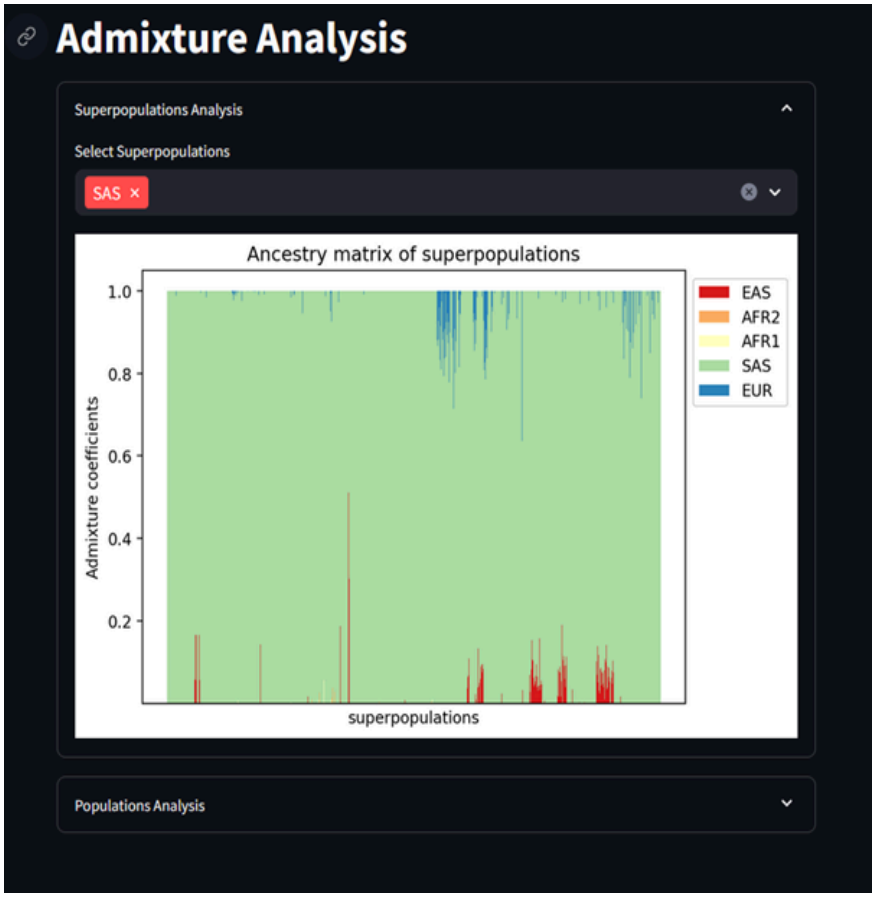

Figure 6. User graphical interface of ADMIXTURE analysis, ancestry matrix of South Asian superpopulation

With the main design objectives in mind (minimalism, error prevention and consistency), this dashboard was designed with the use of Streamlit's expander function to hide/display either analysis ("Superpopulation Analysis" / "Population Analysis") as shown in *figure 6.*

For each Admixture analysis, the user can select as many populations/superpopulation as they wish. This is done using Streamlit's multi-select function, which provides a suitable interface for users to quickly select which groups via a scrolling menu or by typing in the group ID. This method was employed to allow users to quickly interact with the dashboard and prevent errors (for example, a free text box would allow misspellings etc).

As a lot of information may be displayed in any of these Admixture graphs, we incorporated a colour-blind friendly palette as some users may have difficulties identifying single bars of colour within a large display. This could be expanded upon further, by allowing a selection of palettes within the frontend for the user to choose as there are many different types of colour-blindness.

The code for this graph is quite complex and initially took much longer to load. However, several inefficiencies were identified, and alternative code implemented. For example, initially there were several for loops used to isolate specific datasets from the SQL query, this was replaced with one for loop which assigned each row of the SQL query to a specific variable within the script.

As Admixture graphs are stacked bar charts, NumPy arrays were implemented to pass for the “bottom” parameter within the matplotlib’s plot function to improve efficiencies. However, there are still concerns that due to the large volume of data when all five superpopulations are selected via the multi-select function loading times are still significant. This is ameliorated by Streamlit’s built-in loading icon which should make visible the system status.

For future development, it may aid user’s experience by adding superpopulation/population groups along the x axis of the graph for when multiple groups are selected. This can help one’s understanding of the distinct differences between each group.

3.6 Allele/Genotype Frequencies Interface

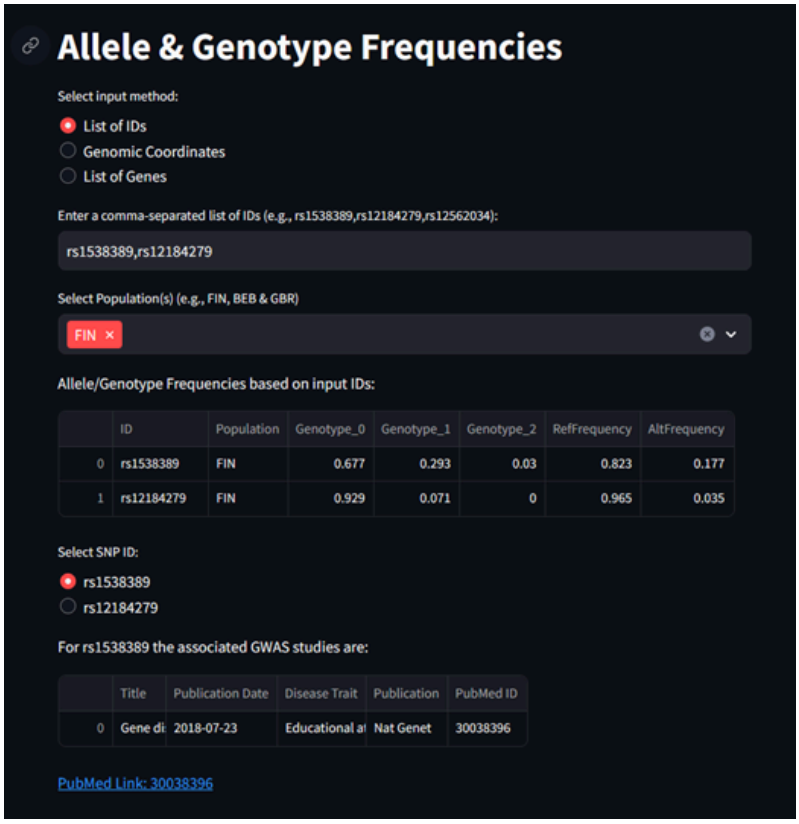

Figure 7. User graphical interface of allele & genotype frequencies page, user selection of IDs and population and, SNPs output and GWAS studies.

As this dashboard has the most options and user input, it was vital during the design process to maintain a simple yet intuitive interface. Examples were provided at all possible times as a method of error prevention as shown in *figure 7*. As discussed in Molich & Nielsen, (1990) clear error messages are helpful, yet error prevention is better for the user's experience which is achieved in this app primarily by providing examples.

As user input for this dashboard may result in large amounts of data to be analysed there are warnings for the "List of IDs" and "List of Genes" paths which encourage a soft cap of 3 IDs or 3 genes respectively but allow a user to input more if they are willing to wait longer.
Figure 7.

This dashboard primarily revolves around an if statement with 3 options in total for the input. The same code is readily employed throughout each 3 journeys with small additional conversion scripts to take the user input and select the relative SNP IDs.
For example, the list of genes provided by the user is checked against our MySQL table named metadata which may produce thousands of SNP IDs depending on the gene. Thus, we allow the user to peruse the relevant SNP IDs and download a csv file of all SNP IDs found within the gene but when returning frequencies only the first 10 SNP IDs are carried forward. If the user has interest in other IDs they can create a list for the "List of IDs" path however, this method may not prove agreeable to users who may prefer to select individual SNP IDs from a gene list to include the frequency table and pairwise calculation.

PandasGWAS (Cao & Huang, 2023) was implemented to check for clinical relevance of SNP IDs using the GWAS catalog data. This package was selected as a better alternative to setting up an API, as an API would require more intricate infrastructure. There was discussion around downloading a list of clinically relevant SNPs found in chromosome one and creating a MySQL database; however, such a table would need to be regularly updated while pandasGWAS is a fully supported python package. In the future, summary statistics could also be displayed within the app through pandasGWAS if deemed beneficial. However, in the future if pandasGWAS was no longer supported and regularly updated it may become incompatible with the GWAS catalog.

The "Genomic Coordinates" option as shown in *figure 7*, does not limit the number of SNP IDs carried forward for the frequency table and pairwise analysis. This potentially could create an issue if the coordinates provided encompass many SNPs, as a large dataset may not be displayed appropriately on the dashboard and the pairwise analysis may take some time to calculate. This could be improved upon by allowing the user to download a csv file of all relevant SNPs and carrying the first 10 SNP IDs forward for the frequency table and pairwise analysis as done for the "List of Genes" option.

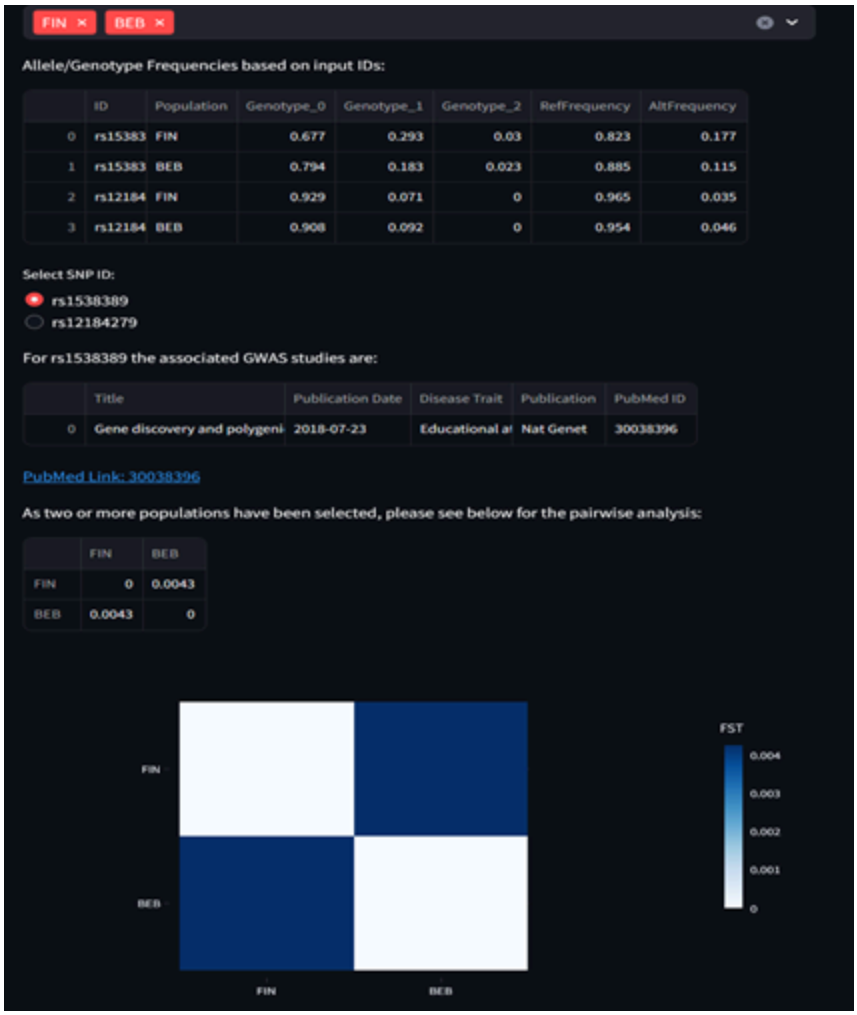

Figure 8. User graphical interface of allele and frequencies page. User selection of SNPs from two or more populations producing heatmap from pairwise analysis.

Currently, if two or more populations are selected the pairwise table and graph are automatically generated towards the bottom of the page (as shown in *figure 8*). As some users may be specifically interested in the pairwise analysis, this dashboard could be improved further by displaying formula information about the pairwise analysis and informing the user that at least two populations will need to be selected before pairwise analysis can be carried out.

4.0 Software Engineering

4.1 Clustering Analysis

For the clustering analysis, PCA was selected as it allows for the dimension reduction of large datasets and can be utilised to observe trends such as genetic similarities across the populations. PCA was chosen over K-means analysis and hierarchical cluster analysis (HCA), as PCA is commonly employed in studies when investigating genomic data and allowed for the reduction of our large dataset (>4000 entries). Although HCA has the same scalability, depending on the parameters used HCA can form clusters even if the data does not inherently have these clusters. Additionally, PCA can easily produce 3D plots which give greater understanding of the data to the user and help prevent bias which may be present in a 2D plot.

The clustering analysis data, like the admixture data, was pre-calculated due to its large size. This allowed for the run time of the software to be reduced in comparison to calculating the data while running the app. Improving the speed and efficiency of the data analysis allows for a better user experience, and a more immersive interface, as populations can be selected within the app for comparison.

To calculate PCA and PVE (proportion of variance explained) from the dataset, PLINK was used in Ubuntu terminal, alongside R Studio (‘pca_code.txt’ on NixOS GitHub). PLINK was used specifically as it also allows for linkage pruning, removal of missing variables and the extraction of data for bed, bim and bam files; files which were utilised in the admixture analysis. In R Studio, the PCA data was finalised and was mapped against the superpopulation and population data in preparation for the user selection aspect of the analysis page. The PCA data contains 20 principal components for each population and superpopulation whilst the PVE has the variance captured for each of these components.

Twenty principal components were calculated for the data as this number captures a significant amount of the variance across the intricate dataset which is displayed in the app for the user’s knowledge (figure 3).

The PCA and PVE tsv files were used to create the corresponding tables within the MySQL database, as seen in figure 2, allowing for this information to be easily extracted by the Django app (figure 5). For the application, the clustering plot was made interactive, users can select individual populations or superpopulations to compare, as well as three components, to understand trends within the data (figure 2, figure 3). Another aspect of the analysis page is that all the graphs and data provided, can be saved by the user, allowing the results to be used in further studies.

4.2 Admixture Analysis

The admixture analysis was conducted with the software ADMIXTURE. This software was chosen in part due to computational limits of hardware requiring the use of the high performance computing cluster (HPC), Apocrita. A job script was required to run the code for the admixture as it would take more resources than an interactive job would permit. ADMIXTURE requires very little code to run, no modules, and needs minimal input for options compared to the admixture software STRUCTURE. These attributes limited the possibilities of an error occurring which would exacerbate time spent in the Apocrita job queue.

Similar to STRUCTURE, ADMIXTURE estimates ancestry of samples by modelling probability of SNP allele frequencies occurring in estimated population ancestries. The ADMIXTURE software differs in its method by employing a maximum likelihood approach. This change in methodology combined with block relaxation and quasi-Newton acceleration has improved ADMIXTURE's speed relative to STRUCTURE.

For the ADMIXTURE code to run, the file labelled admixture within the downloadable ADMIXTURE Linux folder was moved to the same directory as the job script and .bed, .bim, .fam chr1 files produced from previous PCA analysis. The permissions for the chr1 files and admixture file were enabled with the unix chmod command to ensure the files could be interacted with by the job script. The job script was written with the commands to work in the current directory, 6 cores and 6GB of RAM allocated with a maximum runtime of 240 hours. The estimated populations present (k) was input as 5. This is because the sum total of super populations present in the samples is 5 when compared to the 1000 genome project. A cross validation method could have been used to find the optimal k value. Cross validation was not used as time constraints coupled with the information on populations present provided by the sample tsv file meant the method was less attractive, over estimating with the information on hand. The admixture analysis could be improved in the future by running a cross validation. The larger k is the longer an admixture takes, the admixture conducted under these parameters took a day and 20 hours. The true k for regular populations would be much larger and potentially exceed the 10 day limit or require much more resources from the HPC to be requested. Another reason for limiting the number of k to 5 was to reduce the visual noise present on the graph. The number of k is equal to the ancestry coefficient columns produced in the Q file. The percentage of each ancestry coefficient is represented per colour in a stacked bar graph. An increase in k increases the amount of colours needed to represent each ancestry coefficient. Too many ancestry coefficients may be overwhelming for a user to make sense of and reduce the overall contrast of the colours.

The admixture analysis produced a Q file and a P file. The Q file was initially plotted in R as outlined in page 5 of the admixture manual to check the results made sense graphically compared to what we would expect to see. The Q file was reformatted to a csv file and the columns containing sample id and population were added to the csv file from the sample population tsv file supplied. The contents of the csv file and tsv file were easily aligned as they matched by row. The column for super populations was produced by matching the listed population in each entry with the corresponding super population allocated in the 1000 genome project. The ancestry coefficient columns were renamed to the super population each column had the highest values in. European, south Asian and east Asian ancestry all fitted into individual columns but American ancestry was mainly split between east Asian and European ancestry. The final two ancestry coefficients were both related to African ancestry and were

named AFR1 and AFR2. This split was likely due to Africa possessing relatively high genetic diversity relative to other continents and geographical origins of the American populations coupled with the effects of colonialism. The ancestry coefficients were renamed so that users of the app can connect the colours on the graph to associated super populations easier.

The resource strain and time taken for an admixture to be conducted were large motivators for choosing to pre-calculate the admixture to improve efficiency and run-time for users of the app.

4.3 Allele Frequency

Genotype as well as allele frequencies are stored in the database. The values were precalculated to save storage and improve the speed and efficiency of the web app. The user is able to select either a SNP ID, gene name or position range to view genotype and allele frequencies. The user is also prompted to select the population(s) they would like to view. Frequency values as well as SNP IDs and population data are queried from the “freq” table and the selected rows are printed as a table. When position ranges or gene names are used for selection of the rows, the metadata table is used to retrieve the relevant IDs from gene names and positions.

For calculation of genotype and allele frequencies, a csv file retrieved from the original vcf file was used. Using bcftools, the file was filtered and only SNPs which had less than 10% of missing data remained. This seems to be common practice when preprocessing genotype data sourced from the 1000 human genome project. This filtering strategy also improves the quality of the analyses and improves their efficiency. This step drastically reduced the number of SNPs, from 5 million(?) to just under 40,000. In the same step, SNP IDs, Sample IDs and genotype values were retrieved in a csv format.

Thanks to the largely reduced file, further calculations could be performed in R. The file was transformed into a more clean and suitable data frame format. Custom code was used for changing the genotype values to a unified format of 0, 1, 2 or NA when missing (e.g. from ‘0/1’ to ‘1’, suggesting a heterozygote at the locus), removing rows with unknown genotype, truncating unnecessary information from SNP IDs, and other minor changes. A data frame consisting of sample IDs and their populations was merged with the genotype data data frame. Samples grouped by SNP ID, population and genotype were counted and to produce genotype frequencies they were then divided by the number of samples at each locus for each population. Allele frequency data for two alleles - reference and alternative - was calculated from genotype frequencies. A simple equation was used to calculate the reference allele frequencies - the value for the frequency of genotype 0 and value for the frequency of genotype 1 divided by two were summed. This was possible as a locus with genotype 0, which is homozygous, contains two reference alleles, and a locus with genotype 1 (heterozygous) contains 1 reference and 1 alternative allele. An even simpler equation was used to calculate the frequency of alternative alleles: the frequency of the reference allele was subtracted from 1. That is due to the sum of the frequencies having to equal 1. After removing unnecessary columns, the data frame with genotype and allele frequencies was exported as a tsv file to be used for populating the “freq” table in the database.

Gene names were extracted from a chromosome 1 gene list from ENSEMBL and mapped against the chromosome positions in our dataset. This was done in R studio ('genename_code.txt' on NixOS GitHub), using the extracted position columns from the VCF file rather than using the 'bcftools –annotate' package. This was because it was more efficient to map the extracted information, a smaller subset, and add this to our database, rather than load the >3.5GB database into R studio to annotate.

4.4 Pairwise Analysis

To perform pairwise population genetic differentiation, the fixation index (Fst) is used. It is a popular and relatively uncomplicated method of measuring population differentiation due to genetic structure. The equation used is $Fst = \frac{Ht - Hs}{Ht}$, where Ht represents the total genetic diversity in the total population, and Hs is the average genetic diversity within subpopulations. This method is a fairly accurate and efficient way of calculating genetic differentiation between populations. It has certain drawbacks, such as bias when sample sizes between populations vary significantly. However, as the focus of this analysis was efficiency as well as accuracy, this method was chosen over more complicated measures which would affect the runtime of the analysis.

As the user is able to select any number of SNPs and populations to be included in the analysis, calculating and storing the Fst values in advance was not a reasonable possibility. Hence, the calculation takes place once the web app is prompted by the user. The analysis is performed in Python and a function was created which performs the calculation on selected SNPs and populations, and then prints a matrix and a heat map of the results using the package plotly express. Using Python and custom made code as opposed to a package performing population genetics analyses was an optimal approach when prioritising ease of integration with the rest of the web app as well as the simplicity and efficiency of the analysis.

The values in the "pairwise" table in the database are used for the calculation. Using an R data frame created while calculating genotype frequencies, allele counts were calculated. As genotype 0 contains two reference alleles and genotype 1 contains one, the reference allele count for each SNP and population was calculated by multiplying genotype 0 count by 2 and adding the count of genotype 1. The same was done to calculate alternative allele count, except genotype 0 count was replaced with genotype 2 count. Another column containing the total count of alleles was created by multiplying the total count of genotypes by two. A tsv table containing SNP ID, population, total count of alleles, reference allele count and alternative allele count was exported and used to populate the "pairwise" mySQL table. This method of preparation of the values for the Fst analysis was straightforward as previous code was utilised. The function calculating Fst uses the SNP IDs and populations for selection of the values the user chooses, and utilises the reference, alternative and total allele counts for the calculation. In theory, only two out of three of these values are necessary for the analysis, however this design

is optimal for minimising storage needed for the values, while maintaining quick runtime of the analysis.

5.0 References

ADMIXTURE
'This research utilised Queen Mary's Apocrita HPC facility, supported by QMUL Research-IT.' http://doi.org/10.5281/zenodo.438045

Alexander DH, Novembre J, Lange K. Fast model-based estimation of ancestry in unrelated individuals. Genome Res [Internet]. 2009 Sep [cited 2024 Feb 27];19(9):1655. Available from: /pmc/articles/PMC2752134/

ADMIXTURE [Internet]. [cited 2024 Feb 27]. Available from: http://dalexander.github.io/admixture/download.html

Cao, T., Li, A. & Huang, Y. pandasGWAS: a Python package for easy retrieval of GWAS catalog data. BMC Genomics 24, 238 (2023). https://doi.org/10.1186/s12864-023-09340-2

FST
McDonald DB. McDonald's Comparative Physiology Lectures: Fast Start. Available from: https://www.uwyo.edu/dbmcd/popecol/maylects/fst.html. Accessed: [9th February 2024]

Hudson RR, Slatkin M, Maddison WP. Estimation of levels of gene flow from DNA sequence data. *Genetics*. 1992;132(2):583-589. doi:10.1093/genetics/132.2.583

GitHub
https://github.com/GroupNixOS/Project

PCA and PLINK
Speciation Genomics. Speciation Genomics: Principal Component Analysis. Available from: https://speciationgenomics.github.io/pca/. Accessed: [21st January 2024]

Molich, R, Nielsen, J. Improving a human-computer dialogue. Communications of the ACM [Internet]. 1990 March [cited 2024 Feb 27]; 33 (3):338-348. Available from: https://dl.acm.org/doi/10.1145/77481.77486