SNPEuro Portal

Bui V, Gill US, Johnson L & Khairnar N

About SNPEuro

SNPEuro has been created as a web application (web app) designed to give users information on population genomics with specific features. The field of population genomics is the application of genomic technologies to understanding populations of organisms and how genes contribute to health, well-being, and conservation. With recent advances in sequencing technologies, the need for niche platforms into their investigation is underscored.

SNPEuro is a simple user-friendly platform, to allow users to retrieve single nucleotide polymorphism (SNP) information for a genomic region of interest from a single chromosome from all available European population data. Search features of SNPEuro are linked to a curated database with information on the SNPs. The user is able to receive an output for genotype and allele frequencies relative to selected SNPs. They are also able to visualise calculated statistics relative to their chosen SNP, which include a measure of genetic/nucleotide diversity, haplotype diversity and a measure of neutrality. When multiple populations are selected, population genetic variation for each pair value is reported. The user is also able to visualise analysis of the data with the graphical outputs, which can be downloaded for future use and interpretation.

This webapp was developed as part of the MSc Bioinformatics programme at Queen Mary University of London (QMUL), under the supervision of Professor Conrad Bessant and Dr Matteo Fumagalli. Developers of this webapp were V. Bui, U. Gill, L. Johnson, and N. Khairnar. The source code is available on GitHub (https://github.com/MSc-Bioinformatics-Group-Project/SNPEuro).

Table of Contents

1. Design philosophy

SNPEuro Portal was designed to aid biologists with a simple tool of accessing data on population genomics. Inferences from population genomic data are widely used in a number of disciplines, including conservation genetics, evolutionary biology, and precision medicine amongst others. Next-generation sequencing (NGS) technologies have revolutionised the genomics field, providing biologists and researchers alike to attain large amounts of genomic data for many samples from different populations. SNP genotyping is the measurement of genetic variations of SNPs between numbers of a species and the most common type of genetic variation. The analysis of SNPs is important as it is able to link sequence variations to phenotypic changes. Researchers, for example, have found that SNPs may help predict individuals' responses to drugs, susceptibility to environmental factors and the risk of developing certain diseases[1]. Information from SNPs can link sequence variations to phenotypic changes, which can advance and elucidate the molecular basis of disease.

This webapp is a small application to help biologists with limited data retrieval and information regarding population genomics, specifically SNPs, relative to European populations. The webapp is an ideal simple platform for biologists/researchers requiring limited or detailed information regarding SNPs. Our philosophy was to keep SNPEuro simple with a clear purpose, being user friendly with the ability of communicating the information in a structured manner with easy navigation routes. We aimed to maintain consistence with simple colours. The webapp has the potential for scalability in the future. Of key consideration is that this webapp does not rival or replace larger more robust webapps such as Ensemble and UniProt. This webapp is however useful to aid future similar web-based developments in population genomics. We utilised GitHub throughout development to aid version control.

2. Software details

2.1 Architecture

The SNPEuro portal was designed using the Flask framework. A schematic of the software architecture is shown below (Figure 1). In brief this required data collection and curation, building of the webapp along with statistical analysis to be undertaken with integrative capacity. The data used in the webapp was generated using local raw data from the 1000 Genome Project[2], which are stored as VCF files as a database (see section 4; Database details). Statistical tests were undertaken using RStudio and then linked to Flask via the rpy2 bridging package. The main application was run via Flask. Webapp users can access this data via the browser via webpages, displaying data as tables using HTML, which users can download as a CSV file (users can also download the allele and genotype frequency as a .json file for SNP and gene search) or opt for different statistical analysis relevant to the European populations in which they are interested (this is done after the user has entered their input for SNP/gene name or genomic coordinates).

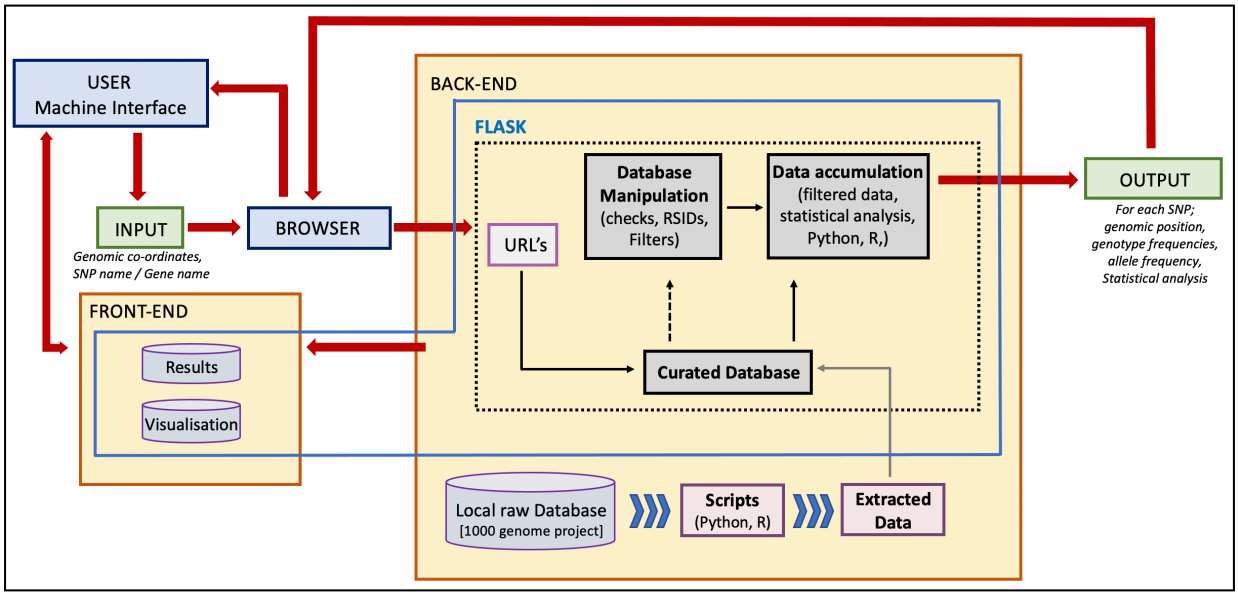

Figure 1: Schematic of software architecture.

The manner in which the software operates is such that, the user enters an input on the webapp, via the browser, this operates to the back-end and feeds the relevant URL. When users enter an input via a form on the webapp on the browser, their input is fed through to the back-end and the input is displayed through the relevant URL. At the back-end this will 'access' the curated data files which have been generated. The data will then be manipulated/accumulated relevant to the input from the user. The required information, as per request will then be delivered as an output to the user, via the browser and user interface. The user is able to access the webapp, with the specific features created for the front-end, through the landing page, navigation bar and footer, allowing for generation and visualisation of results.

2.2 Software & Webapp organisation

The integration of the various components of the software to make software schema is outlined in Figure 2. This shows the connection of the database with the main application. Data for the database was extracted from dbSNP via NCBI. Indicated below are the fields of the database. The database was integrated with the main application which was generated using Flask as a website developer platform. Statistical analysis was performed with RStudio which linked both with the database and the main application. HTML and Cascading Style Sheets (CSS) was used to produce the front-end of the main application. HTML was used to produce templates for the web pages of respected routes made using Flask. CSS was then used to style and enhance the HTML templates.

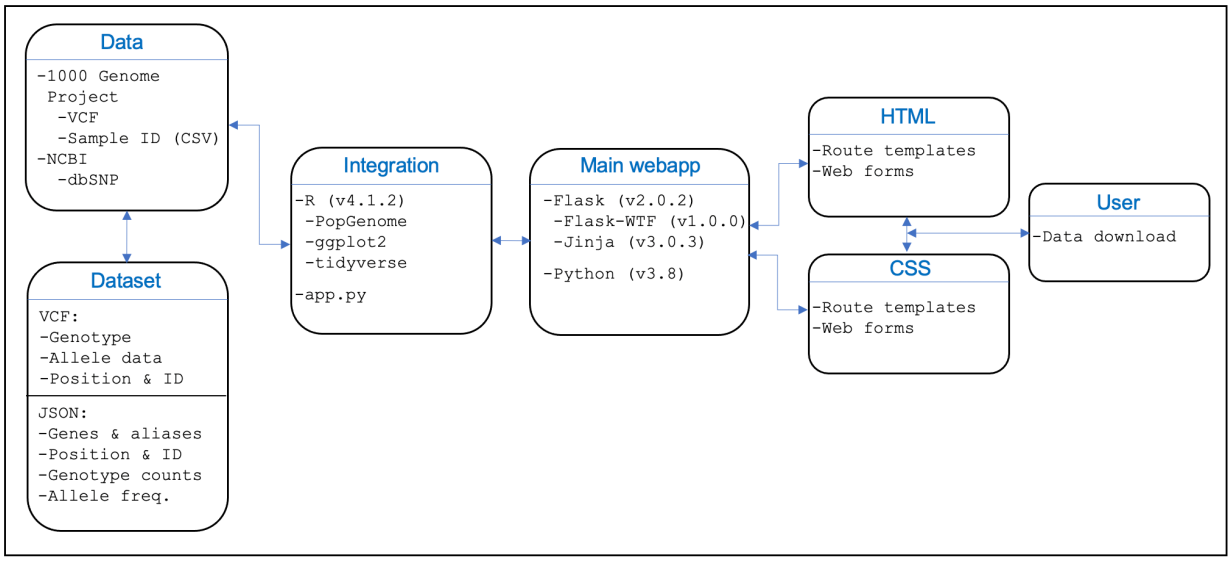

Figure 2: Schematic of software integration with different components

A schematic of the webapp process and organisation is shown Figure 3, (main specific features of the webapp are discussed in section 5). This outlines the different hyperlinks that are available to the user and their respective outputs.

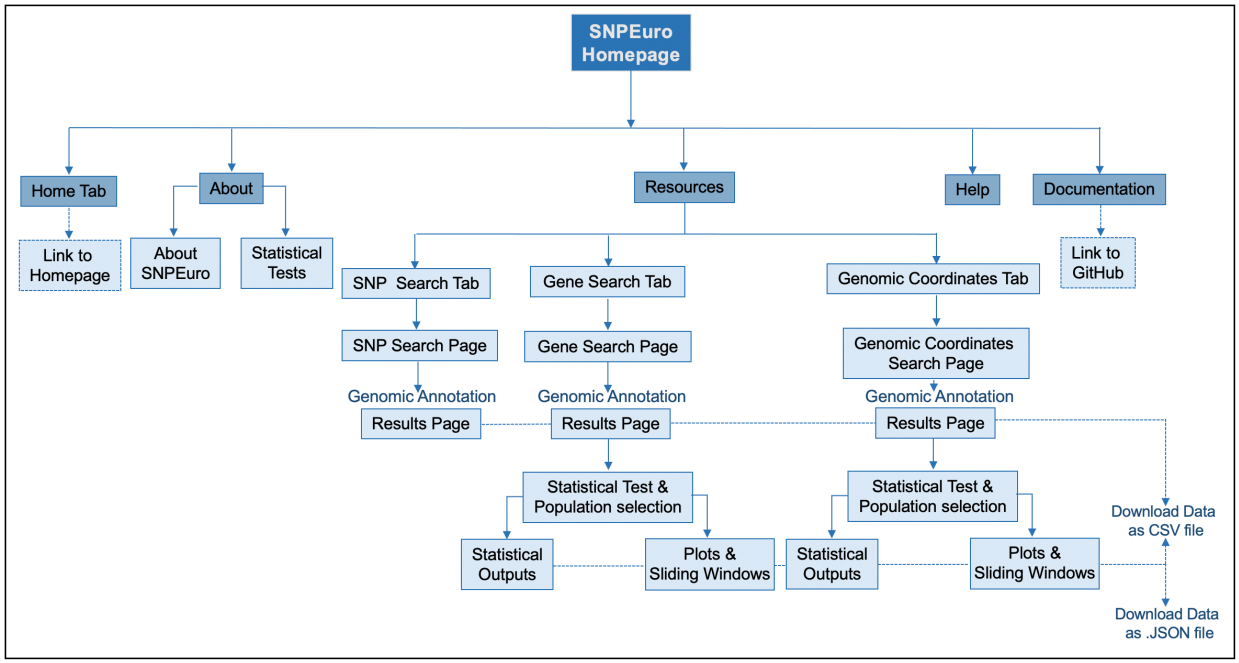

Figure 3: Flowchart of web app

The SNPEuro app includes a home/landing page which provides the type of searches the webapp offers with appropriate hyperlinks and graphics. Users can also access specific internal hyperlinks as they navigate through the webapp via the navigation bar or footer. An 'About' dropdown tab includes an 'About' page detailing the utility of SNPEuro and also a 'Statistical Tests' page, which includes details about the statistical tests which have been utilised within SNPEuro. Further single hyperlinks including a 'Help' and 'Documentation' page with an external link to GitHub are also included. A 'Resources' dropdown tab is offered which links to the tools offered by SNPEuro (see section 5; Webapp features). This includes search windows for SNP (rsID), gene search and

genomic co-ordinates (positions). The webapp returns, for each SNP, genomic position, gene name, the genotype frequencies and allele frequency, where frequencies are provided for each population separately. For gene search and genomic coordinates search, users have to download a file containing the genotype frequencies and allele frequencies.

When multiple SNPs are returned the user is able to select the European populations of choice; British, CEPH (Centre d´Etude du Polymorphisme Humain), Finnish, Iberian and Toscani. Users can opt for statistical analyses if multiple SNPs are shown. These can be shown dependent on the different European populations selected and the desired statistical tests from the user (see section 6; Statistical Analysis section). The selection of both functions for populations and statistical tests asks the user to specify by typing, their desired selection. We opted for this over a drop-down menu for the ease of selecting multiple populations and/or statistical tests simultaneously. Graphical representations are presented in accordance with the chosen statistical test by the user, either as separate graphs or sliding windows according to the range selected by the user. Summary statistics can also be downloaded as a csv/txt file, which is provided as a feature on the webapp. Users can also have the option to download the results page before the statistical outputs as we have included a download button at that position

3. Technologies

Technologies utilised for the creation of SNPEuro included the Flask framework using the Python language. Data curation using BCFtools[3] with data collection via the 1000 Genome Project[2], as described below. We employed RStudio for statistical analysis with the rpy2 platform as a bridge linking R with Python. For download purposes .JSON and CSV files have been used. Specific technologies have been described in the sections below relevant to the software architecture.

4. Database Details

4.1 Data collection & curation

Primary data collection for the webapp was obtained via use of the 1000 Genome Project[2] website/tool (https://www.internationalgenome.org). Data pertaining to the full chromosome 21 was undertaken with all data from the 30X study[4], as VCF genotype/haplotype information for each sequenced sample at each SNP from all populations. Full chromosome data, (rsID numbers, genes, psotions) were extracted from the dbSNP via the NCBI website with all permutations. Via bash command, BCFtools were used to filter the VCF for the required populations relevant to SNPEuro. This was annotated with rsIDs with BCFtools using complete dbSNP VCF provided by NCBI (00-All.vcf.gz on this link https://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/VCF/). Extraction of rsID positions for the reference and alternative allele from the now annotated 1000 Genome Project, and this was then stored as a .txt file. Plink2[5] was the used on this annotated VCF to calculate allele frequencies and genotype counts for each position in the 1000 Genome Project VCF. Returned results show the position and rsID (if available), which are stored in a .txt file. Using the rsID, positions, reference and alternative allele .txt file from above, a dictionary was created in Python with rsID as the key and the remaining parameters as the values. Respective genotype and allele frequencies were then added to this dictionary (NB those without rsIDs are not stored). The output here is a .JSON file for storage, which is read each time as required. For example, if the rsID was 123, then the following:

rs123”:[100,’a’,’g’,0.00001,’0/0=3,0/1=4,1/1=100’]
Position, Reference, Alternative, Allele Frequencies, Genotype counts.
For gene name, this is used as the key, with the positions of the rsIDs associated as the values, then taking the maximum and minimum for those position for the requested input for reading the readVCF function in R:
i.e., "Gene1":[1,3,5,17,66,153] -> readVCF(1,153)

Once the data was collected into the VCF file format, downstream reading was undertaken for statistical analysis using an R package; PopGenome[6] (version 2.7.5). The VCF file was read using the `readVCF()` function, which requires a vcf.gz, compressed file, and its indexed file in the vcf.gz.tbi format, stored in the `“GENOME.class”` object. Subsequent data curation for statistical analysis was undertaken in RStudio using the PopGenome package (see section 6; Statistical analysis). Consequently, to integrate the R-based scripts for statistical analysis into Flask, using Python, we used the rpy-2 package, developed specifically as a Python-R bridge. This therefore allows the user input to change relevant request information within the Flask framework. For the purposes of graphical outputs being merged onto the Flask framework, we saved these outputs to the hard drive as a .jpg file and then used that .jpg file to load into Python.

4.2 Database manipulation & wrangling

For database manipulation, which included data generation (e.g., allele frequencies and genotype counts) for which we used Plink2 and BCFTools. Data wrangling was undertaken with BCFTools and easy.entrez and files were stored as .JSON and compressed .vcf.gz. Although there are a number of other platforms which could have been used for database handling and manipulation, such as Structured Query Language (SQL), we decided that these were not essential for this webapp. We required access, manipulation and communication with the database, and for statistical analysis in R, VCF files were required and thus used as noted above. For access to all gene names and associated aliases, within R we employed the ‘org.Hs.eg.db’ package (via Bioconductor). Whilst we could have undertaken this using SQL for allele frequencies and counts which has the advantages of faster query processing and the limited need of coding skills with multiple data views it still has a complex interface and for the purposes of our simple webapp we felt its use was not required.

5. Webapp features

The main features of the SNPEuro webapp were to provide information and statistical analysis of SNPs in relation to the specific European populations selected. The main downstream searches were then undertaken using the ‘Resources’ internal hyperlink/tab. From this link differential searches were made possible, with further outputs of the potential to request data relevant to the European populations and statistical analyses.

5.1 rsID search

The user is able to enter a specific SNP, with an rsID, to obtain information with regards to the genomic position, genotype and allele frequencies. The user essentially enters a string with the readout being alike. It is evident that those biologists entering specific SNP details will have an in-depth knowledge of their search, which will then reveal the relevant outputs for them. The outputs include the genomic position of the SNP, biallelic and polyallelic frequencies along with the nucleotide diversity.

5.2 Gene search

For the gene search internal hyperlink, the user is able to enter the name of a gene or any alias of it to present the relevant genomic data, as outlined above. This is also entered as a string with the similar output. Here multiple SNPs may be returned relative to the gene ID or gene alias, with which the user is able to select specific statistical tests and summary statistics can be delivered. Gene searches may be used by biologists or researchers who want to know the specifics of a gene, not knowing about positions or rsIDs, thus the webapp is useful for both scientists that have knowledge relative to the SNPs and those who have limited knowledge and require further information and detail.

5.3 Genomic coordinate search

Similar to the gene ID/alias search, the user is able to select the SNP position/genomic coordinates, following which all SNPs in relation to the range of coordinates is delivered and users can then select relevant statistical tests they wish to see. The user enters an integer as an input here, where the readout is displayed as a string. A limitation of our webapp is that due to the RStudio technology, we had to cap the range for the genomic co-ordinates, such that the range could not excel 140,000.

5.4 Statistical test selection

As multiple SNPs are returned from the aforementioned searches, users are able to select specific statistical tests, which they enter in a text box. Nucleotide diversity is already returned on the main results page. Users are able to then select further diversity statistics by inserting haplotype diversity, d_Xy, which generates the absolute nucleotide divergence between 2 populations. Neutrality measured with Tajima's D, can also be selected. Users can then select a range for a sliding window to allow for the statistical data to be visualised.

5.5 Population selection

Along with the features of selecting the genomic variables and statistical analysis, users are also able to select their European populations of interest relative to the genomic variables. Users can select the specific populations in which they are interested by inserting this into the text box, where more than one population is separated by a comma. Summary statistics are then generated according to user selection along with the corresponding visualisations.

5.6 Downloading data

The user is finally able to download the genomic annotation results and statistical summary data in a CSV format along with the graphical outputs also available as .JSON files, to use for further purposes and interpretation as required for their desired research work or otherwise.

6. Statistical Analysis

6.1 Statistical Test development

Within the SNPEuro Portal webapp, following retrieval of information on the relevant SNPs, when multiple hits are returned, specific to populations, selected summary statistics can be calculated. Statistical analysis which is delivered by SNPEuro includes nucleotide diversity as a measure of genetic diversity (denoted as Pi), haplotype diversity, d_Xy, which is a measure of absolute nucleotide divergence between two populations. In addition, when multiple populations are selected then population genetic variation (F_{ST} value) for each pair of

populations is reported. Tajima's D as a measure of neutrality is also calculated. Using the PopGenome package in RStudio all statistical analyses could be undertaken within a single platform. Here we describe the selection of statistical tests that are utilised within the webapp and then the relevant functions from the PopGenome package which were used to implement and employ these for data generation

Genetic Diversity (Nucleotide & Haplotype)

For statistical tests we selected diversity statistic measurements which included nucleotide diversity and haplotype diversity. Nucleotide diversity is a concept in molecular genetics measuring the degree of polymorphism within a population. Nei and Li in 1979 first introduced the measure of nucleotide diversity[7], which is simply defined as the average number of nucleotide differences per site between two DNA sequences in all possible pairs within the same population, (this is denoted as Pi). Nucleotide diversity if a measure of genetic diversity and commonly used in combination with other statistical analyses of population genomics and diversity and is similar to expected heterozygosity, a measure of haplotype diversity[8,9]. Within populations highly diverse or well-balanced libraries have approximately equal populations of all four nucleotides in each cycle throughout the sequencing run. Conversely, low diversity libraries have a high proportion of certain nucleotides in a cycle[10,11]. Within our SNPEuro portal we are able to provide nucleotide diversity between SNPs comparing different populations, denoted as Pi within the results table. Also, a measure a genetic diversity, our webapp also provides a calculation for haplotype diversity. A haplotype is a set of DNA polymorphisms that tend to be inherited together and can be a combination of alleles or set of SNPs found on the same chromosome. Haplotype diversity represents the probability that two randomly sampled alleles are different and can be implicated by a number of different processes such as mutation, recombination, marker ascertainment and demography[12]. It is thus the measure of the uniqueness of a particular haplotype in a given population[13]. Users of SNPEuro can select for this statistic to be read from their chosen genomic and/or population searches.

Genetic Differentiation & Divergence

Within the webapp the user can select different European populations as noted above and then population genetic variation is delivered. This measure, also called the fixation index (F_{ST}). F_{ST} is denoted as the proportion of the total genetic variance contained in a subpopulation relative to the total genetic variance and thus is a measure of population differentiation due to genetic structure, thus a measure of genetic differentiation and when species diverge in the presence of gene flow[14]. It is commonly estimated from genetic polymorphism data (SNPs) as it is within this webapp. Population selection strongly influences F_{ST} values, such that closely related groups are indistinguishable, displaying panmixis, where 2 populations are interbreeding freely[15]. Thus, within this SNPEuro webapp, expectedly there may be low F_{ST} values between the selected populations as the Europeans populations may well be interlinked. A third statistic within genetic diversity, (d_XY), which is the absolute nucleotide divergence between two populations is also provided. This measure showing the patterns of genetic divergence between populations as a function of nucleotide diversity revealing differential gene flow during speciation[16].

Neutrality

In line with genetic diversity, key in population genomics is also a measure of neutrality. Neutrality tests compare two estimators of the population mutation parameter that characterises the mutation-drift equilibrium. The Tajima's D test is one measure of neutrality and is computed as the difference between two measures of genetic diversity[17]. It is the mean

number of pairwise differences and the number of segregating sites, each scaled such that they are expected to be the same in neutrally evolving population of constant size. Tajima's D thus distinguishes between a DNA sequence evolving randomly (or neutrally) and one evolving under a non-random process. A negative result signifies an excess of low frequency polymorphisms relative to expectation (population size expansion), whereas a positive result signifies low levels of both low and high frequency polymorphisms, indicating a contraction in population size and/or balancing selection[18].

The SNPEuro app delivers the statistical outcomes, relative to the tests indicated above along with graphical visualisations. The readouts utilised here include a sliding window and plot for Tajima's D. The sliding window analysis is a method studying the properties of molecular sequences, where data are plotted as moving averages of the criterion in question (e.g., nucleotide diversity), for a window of a certain length slid along a sequence or sequence alignment.

6.2 PopGenome package

To undertake the statistical analysis within the SNPEuro webapp the PopGenome package within RStudio was utilised. This provides sufficient tools for the appropriate genomics data analysis. We noted this package included a wide range of polymorphism and neutrality tests and FST estimates with suitable execution from sequence data from the 1000 Genome project similar to that for SNPEuro. The package allows for the full range of methods to be applied to whole alignments, sub-sets of sequences and sliding windows as we have utilised, based on nucleotide positions or on SNP counts.

6.3 Statistical analysis code outline

Following installation of the PopGenome package, statistical analysis was carried subsequent to loading the relevant libraries.

```
Library (tidyverse)
Library (PopGenome)
Library (ggplot2)
Library (bigmemory)
```

Population data was set and then position data was obtained, with the `GENOME.class` object, which provides information about the main methods provided by PopGenome. This object/function is also used to read the VCF file with the information from the selected populations for SNPEuro. Initially SNP counts are obtained. The sliding window is then generated, which is a function contained in PopGenome. The function `sliding.window.transform()` transforms an object of `class.GENOME` into another object of `class.GENOME`, where new regions correspond to individual windows, specified by the user. This mechanism enables the user to apply all methods that exist in the PopGenome framework. PopGenome concatenates the data if the parameter `whole.data=TRUE`. This mechanism enables the user to work with very large datasets, which can be split into smaller chunks that are stored in the input folder. PopGenome is able to concatenate these chunks for analysis. The function used; `readVCF` undertake this automatically. If `whole.data=FALSE`, the regions are scanned separately, which can be done as type=1: Define windows based on SNP counts or type=2: Define windows based on nucleotide counts.

The `diversity.stats-methods` were used as a generic function to calculate nucleotide and haplotype diversities, which require division by the slot `GENOME.class@n.sites` to obtain

diversity per site. Subsequently F_{ST} statistics and *d*_XY (diversity stats) can be obtained, where the latter rely on the former, in relation to the populations. Neutrality (Tajima's D) is then calculated with the `neutrality.stats` function.

6.4 Visualisation of statistical outputs

Following calculation of summary statistics users are able to visualise the results of aforesaid tests with graphical representation, including plots within a sliding window for each of the statistical tests. The user is able to select the range of the sliding window on the landing page which then depicts the image which is displayed. This, however, is a limitation of the visualisation, where the output of the sliding window would ideally have been dynamic. However, with the transfer of RStudio code to Flask using rpy2, this proved difficult and thus images required to be saved as PDF or JPG format rather than within a dynamic scale. Users are still able to visualise comparisons between populations and of course can alter the sliding window range as required.

6.5 PopGenome; Pros & Cons

We decided to use the PopGenome package, which can efficiently process genome-scale data as well as large datasets via the R platform, thus utilising different bioinfomatic/code packages as taught. A number of computer programs exist for performing population genetics calculations, they can be limited in the analyses they provide within a single platform, which was an advantage of PopGenome. The package can read associated annotation files in GFF format, which enables the easy definition and classification of SNPs based on their annotation, with analyses applied to sliding windows as depicted in SNPEuro. The diverse statistical analysis offered by this package was an attractive attribute for its use. The integration using R facilities with downstream analysis allowed the production of high-quality graphics, which were limited with other packages that we had explored. The limitation we noted with using PopGenome and thus RStudio was the additional step of integrating these analyses from R to Python for the Flask framework. We utilised rpy2 for this purpose, but this was an additional step, which would not have been required had a Python related package been used, however, we felt the graphical readouts using PopGenome in RStudio were superior to those in other packages explored. In RStudio, we did also note a technical issue where we were not able to select a genomic coordinate with a range of >140000 and thus have included this as a stipulation in our webapp. This, of course, is not ideal and a limitation of using RStudio here, as any value with a range larger than this led to abortion of the package and thus unable to execute the code any further. We experimented by changing from Linux to Mac OS operating machines but were unable to solve this issue. Currently we have stipulated this as the maximum range for genomic coordinate insertion and would improve this feature should we have had more time and in future development.

7. Flask

7.1 Flask architecture

The Flask framework depends on the Jinja template engine and the Werkzeus WSGI toolkit. It has an easy link integrated with Python. Flask runs with Python version 3.6 or newer and is based on Jinja 2 sharing a similar language to Python, with easy integration to run code on the webapp as well as HTML. Within the framework we have a main homepage, which includes the searches

available with the relevant links. A navigation bar includes the tabs as denoted above, which the user can click on. A documentation tab also links to the GitHub repository. The homepage allows the user to select the type of search they desire which redirects them to an appropriate search page where they can enter different SNP detailed information – allocated to different tabs (e.g., genomic co-ordinates etc). The packages that are required for the installation are in the app2.py folder (please also see requirements.txt file)

7.2 Flask Framework

The information provided here is relevant to the developer of the webapp. We utilised the Flask framework for creation of the webapp. Additional technologies that were required for integration with Flask included our curated database with RStudio and rpy2.

For the installation of Flask a virtual environment is created within a folder (using a python writer; we opted for Visual Studio Code) from where the developer wishes to run their framework.

```
pip install virtualenv
Virtualenv env
```

The virtual environment is then activated prior to initiating the framework, with the following code:

```
Source venv/bin/activate
```

The virtual environments allow the developers to manage the dependencies for the project, both in development and production. These environments are independent groups of Python libraries and thus one project does not affect other projects or operating system packages. Using a virtual environment also makes it simpler to track packages used. The packages used are indicated in the requirements.txt file

Within the activated environment, Flask is installed via the command line:

```
pip3 install flask
```

An SNPEuro folder is then created, with an `app2.py` file, which is the main constituent containing the Flask application. The relevant code for this set-up is:

- Import Flask: `(from flask import Flask)`
- Set-up application: `(app=Flask(_name_)`
- Set-up route of homepage: `@app.route('/")`- [within the brackets included is the URL setting].
- Define function for the route, called `index()` in this case: `Def index():`
- Set-up the route for the 'About', 'Statistical tests', 'Help', 'Gene search', 'SNP search', and 'Genomic co-ordinate'.

HTML templates are then produced within the Flask framework which were enhanced using CSS. Furthermore, bootstrap was used for website delivery, which included the navigation bar with drop down options for the resources tabs including gene ID, coordinates, and SNP searches (achieved through bootstrap). A footer was also created which also included the links on the navigation bar.

The Flask web forms were created with various libraries. Flask-WTF was installed for working with forms within the framework:

```
pip install Flask-WTF
```

and the following code used for library imports:

```
from flask_wtf import FlaskForm, import Form
```

```
from wtforms import StringField, SubmitField
from wtforms.validators import Required
```

The created webforms, noted above, were then linked for the statistical analysis. R packages were imported, with `rpy2.robjects as robjects`. A class was created to define a form for SNP (SNPForm) along with a search for the SNP. The data was extracted from a .JSON file. Similarly, forms were created for gene name/ID and genomic coordinates. Here we included the relevant return information should specific SNPs, genes, co-ordinates not be available, with the generation of code using exceptions. To test that the variables entered in each form were functional, we used truncated data (VCF files) to determine if the correct outputs were delivered and adjusted the code accordingly in the event of errors arising. Having imported the rpy2 package as the bridge between Python-R, we were able to integrate the code from RStudio for the statistical analysis to Flask. Graphical outputs of the statistical analyses, which includes sliding windows, were also integrated within the Flask framework.

7.3 Front-end & back-end framework

Development of the front-end of the SNPEuro webapp was largely performed using HTML and CSS – with bootstrap which allowed for the improvement in functionality of the webapp. This included the creation of various tabs, a footer page and the inclusion of data entry points for the user. The different views from tab or selection otherwise are then returned to the user as a HTML template, which contains the information regarding the manner in which the information is to be displayed to the user via the browser. We note that in hindsight and with additional time we would also provide additional focus to the front-end with further bootstrap use to improve the SNPEuro webapp in terms of aesthetics, but the priority here was to complete a functioning prototype with the software requirements as requested.

The table below (Table 1), in conjunction with Figure 3, includes the relevant tabs that are included with the webapp with a brief description of their function for the front-end of the webapp.

HTML/CSS	Page Function
main.css	Organises orientation of webapp
home.html	Content for homepage
base.html	HTML base, which subsequently extends to other pages
about.html	Content for about page
about_statistics.html	Content regarding the statistical tests used
coordinate_result.html	Statistical outputs for specified genomic coordinates
gene_view.html	Content for gene ID or alias
gene_result.html	Statistical outputs for specified gene
genomic_view.html	Content for start, end genomic and results page
help.html	Content for help page
index_gene.html	Form for gene name
index_genomic.html	Form for genomic coordinate input
index_page.html	Form for SNP input
result.html	Contains results for statistical analysis & sliding window/plots
rsif_data.html	Contains gene annotation for SNP search

Table 1: indication of HTML pages/templates for the site layout

7.4 Flask; Pros and cons

There are number of technology platforms that can be used for webapp software development such as Flask, Django and FastAPI, which are operated with Python. We opted to use the Flask framework, with it being flexible, as most parts of Flask have the possibility of changing, which is not necessarily the case for other platforms. It has compatibility with the latest technologies and was simple enough to integrate with R via the rpy2 package. Flask is also beginner friendly due to its overall simplicity and enables app development with ease. In addition, it is easy to experiment with Flask with the overall architecture and libraries, especially as we were not initially certain which libraries or frameworks would work best for our webapp. The SNPEuro webapp is a small platform, and we noted that when developing the app further Flask may not be the best platform, though for prototype purposes it seemed ideal as it is known to be well utilised for simple cases and web applications. Flask also is easy for documentation purposes, with tips arranged in a structured manner.

For the purposes of this small webapp, the disadvantages of Flask were limited, but potentially include risks for security, with slower development and complicated maintenance for larger implications, factors which would need to be considered. Furthermore, for future use, Flask lacks a large toolbox, thus developers need to manually add extensions such as libraries. The addition of a number of extensions may lead to the app becoming slow. The modular nature of the Flask framework means that many developers working on software together need to familiarise themselves with each constituent part of the framework.

8. Limitations & Technical Solutions

There are a number of limitations which we encountered in the development of SNPEuro, some of which we were able to offer technical solutions, which are discussed below. We have differentiated these into those related to database design and manipulation, statistical analysis and that related to the webapp framework.

8.1 Dataset

We extracted from the complete dbSNP file, which is a large sized file, this created issues with the handling of commands with the webapp. The advantage of this was that all possible rsIDs were included for a complete dataset. The VCF files remain too large to facilitate searches above 140,000 base pairs and would require truncating and re-concatenating to allow this limitation to be overcome. From the easy.entrez package, rsIDs are extracted which do not coincide with the dbSNP data, where it appears that the position is incorrect by a factor (of millions) for a few rsIDs. This would mean that these genes would span an extremely large range, which is likely not possible. We therefore required manual checking of these. This could also be averted by validating against other databases. For the separate populations, the genotype counts, and allele frequencies have some duplicated data across the .JSON files. This could be overcome by having a single file, time-permitting to create this in the future.

8.2 Statistical analysis

Despite opting for a statistical package in RStudio, which entailed all statistical analyses to be undertaken within a single platform with suitable functions for plots, these were not interactive. It is likely that this issue was due to the complexity of the bridging between R and Python via rpy2. To meet the said requirements, we opted graphical visualisations were imported as JPG

files, but these would be made interactive in future development. Given more time, this could be possible using R with the plotly package. As PopGenome contains a number of additional statistical tests these could also be included within the webapp for future development, providing a more robust analysis of population genomics.

8.3 Webapp Framework

We note a number of limitations within our SNPEuro webapp, some of which have been discussed above. The aesthetics of our webapp need improving so that it is consistent and engaging with the user. As technical solutions to improve the usability of the webapp we would opt for tab selection and/or drop-down menus for selection of statistical tests and/or populations prior to delivering the outputs of the data. As noted above with the statistical analysis, we would also provide an interactive sliding window. We note an issue with the range for the sliding window, which the user inputs. As a future technical solution, we would hard code an upper and lower limit such that the user does not require filling in this parameter, thus improving usability. We would also include a separate generic search bar where the user can search the entire data with the correct input. In addition, external link to publications or other databases (Ensemble, NCBI) would be created relevant to the gene or SNP that is of interest to the user.

9. Future Development

Future developments for SNPEuro would be to adapt the database, changing from uploading the database directly, to this being hosted by Pandas or SQL with a server and with automatic real time database updates for the SNP data, making it more relevant and applicable to the user. As previously noted, we would improve the aesthetics and functionality of the webapp using drop down and selection tabs as opposed to text boxes. In addition to downloading .JSON and CSV files for the gene and genomic coordinate search we would instead create an instant results table with this information. Further interaction within the webapp would be created by adding other visualisations including SNP networks, sliding window circular plots and t-distributed stochastic neighbour embedding (tSNE) plots. To further advance the webapp we would also have specific external hyperlinks to gene names, SNPs directing users to Ensemble and/or specific literature relevant to the user's search. The webapp would also be improved by adding error messages when the users request is not met.

10. Deployment

To allow the software to function on a target device the webapp could be deployed. Amazon web services (AWS) offer methods of deployment tools which provides a fully managed deployment service that automates software deployment. Elastic beanstalk is a platform within AWS that is used for deploying and scaling web applications with a number of different languages, including Python, which is what we would use for deployment. Elastic beanstalk is known to be a simple platform where code can easily be uploaded, and it handles the deployment. It is important to ensure that the functionality of the software is smooth and determine the URLs are correct. For further deployment a requirements file would be required so that the correct packages can be operated. We would further explore the exploration of SNPEuro as a future consideration.

11. References

1. Lander, E.S.*, et al.* Initial sequencing and analysis of the human genome. *Nature* **409**, 860-921 (2001).
2. https://www.internationalgenome.org.
3. Danecek, P.*, et al.* Twelve years of SAMtools and BCFtools. *Gigascience* **10**(2021).
4. Marta Byrska-Bishop, U.S.E., Xuefang Zhao, Anna O. Basile, Haley J. Abel, Allison A. Regier, André Corvelo, Wayne E. Clarke, Rajeeva Musunuri, Kshithija Nagulapalli, Susan Fairley, Alexi Runnels, Lara Winterkorn, Ernesto Lowy-Gallego, The Human Genome Structural Variation Consortium, Paul Flicek, Soren Germer, Harrison Brand, View ORCID ProfileIra M. Hall, Michael E. Talkowski, View ORCID ProfileGiuseppe Narzisi, Michael C. Zody. High coverage whole genome sequencing of the expanded 1000 Genomes Project cohort including 602 trios. *bioRxiv* (2021).
5. Chang, C.C.*, et al.* Second-generation PLINK: rising to the challenge of larger and richer datasets. *Gigascience* **4**, 7 (2015).
6. Pfeifer, B., Wittelsburger, U., Ramos-Onsins, S.E. & Lercher, M.J. PopGenome: an efficient Swiss army knife for population genomic analyses in R. *Mol Biol Evol* **31**, 1929-1936 (2014).
7. Nei, M. & Li, W.H. Mathematical model for studying genetic variation in terms of restriction endonucleases. *Proc Natl Acad Sci U S A* **76**, 5269-5273 (1979).
8. Goodall-Copestake, W.P., Tarling, G.A. & Murphy, E.J. On the comparison of population-level estimates of haplotype and nucleotide diversity: a case study using the gene cox1 in animals. *Heredity (Edinb)* **109**, 50-56 (2012).
9. Stumpf, M.P. Haplotype diversity and SNP frequency dependence in the description of genetic variation. *Eur J Hum Genet* **12**, 469-477 (2004).
10. Booker, T.R. & Keightley, P.D. Understanding the Factors That Shape Patterns of Nucleotide Diversity in the House Mouse Genome. *Mol Biol Evol* **35**, 2971-2988 (2018).
11. Garcia, R.J., Cox, M.P. & Hayman, D.T.S. Comparative genetic diversity of Cryptosporidium species causing human infections. *Parasitology* **147**, 1532-1537 (2020).
12. Cagliani, R.*, et al.* A positively selected APOBEC3H haplotype is associated with natural resistance to HIV-1 infection. *Evolution* **65**, 3311-3322 (2011).
13. Buntjer, J.B., Sorensen, A.P. & Peleman, J.D. Haplotype diversity: the link between statistical and biological association. *Trends Plant Sci* **10**, 466-471 (2005).
14. Willing, E.M., Dreyer, C. & van Oosterhout, C. Estimates of genetic differentiation measured by F(ST) do not necessarily require large sample sizes when using many SNP markers. *PLoS One* **7**, e42649 (2012).
15. Bhatia, G., Patterson, N., Sankararaman, S. & Price, A.L. Estimating and interpreting FST: the impact of rare variants. *Genome Res* **23**, 1514-1521 (2013).
16. Crawford, J.E.*, et al.* Reticulate Speciation and Barriers to Introgression in the Anopheles gambiae Species Complex. *Genome Biol Evol* **7**, 3116-3131 (2015).
17. Tajima, F. Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics* **123**, 585-595 (1989).
18. Cagliani, R.*, et al.* Balancing selection is common in the extended MHC region but most alleles with opposite risk profile for autoimmune diseases are neutrally evolving. *BMC Evol Biol* **11**, 171 (2011).