BIO727P SOFTWARE GROUP PROJECT

Software Documentation – SNP Web Application

Amanah Lewis-Wade[1*], Celine Lie[1] and Gracia Andriamiadana[1,2]

*Correspondence:
a.lewis-wade@se21.qmul.ac.uk
[1]Queen Mary University of London, Mile End Road, E1 4NS London, UK
Full list of author information is available at the end of the article

Project Philosophy

This web application was designed to allow bio scientists to retrieve information on SNPs and calculate simple genetic population genomic statistics of their choice within a downloadable text file and visualise the distribution. SNP ID, gene names, gene alias, and chromosomal region of interest can be entered in the search bar and these search terms will be checked off in the form below. The server will retrieve either single or multiple SNPs depending on the requested input, which could be associated with many SNPs. Furthermore, chromosomal coordinates, derived allele frequency, allele and genotype frequencies for each population corresponding to each SNP would be retrieved. The user can select one or more summary statistics of interest and a number of populations. For storage reasons and due to the short amount of time of this software project, the available summary statistics are Shannon's Diversity, Expected Heterozygosity, and Tajima's D, while the five populations that can be chosen from are British, Dai Chinese, Gujarati, Luhya, and Mexican. When there is more than one population requested by a user, a FST analysis can additionally be done with the option to download the selected stats in a text file. The workflow and structure of the web application is shown in Figure 1: Collecting the SNP VCF file, annotation, population (meta) data from from Ensembl Genome Browser FTP Server, and gene alias from `R Bioconductor`, all data sets were processed using different tools and packages. The database was built based on the processed data using `SQLite3` and store as a .db file. The website is linked to the database through the web development frame work `Flask`. On the website, users can input rs ID, gene name (or alias), and genomic coordinates, and perform the above described analyses.

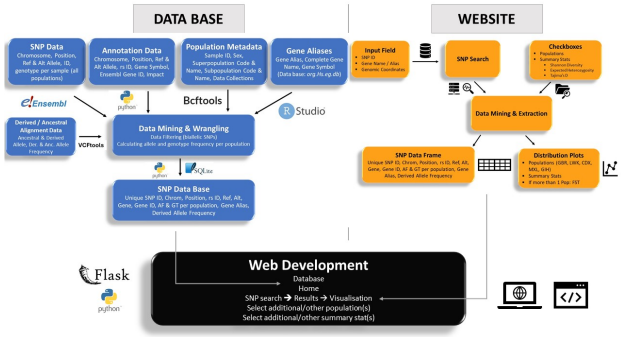

Figure 1: **Workflow and Structure of Web Application Project.** The left side shows the workflow of building the database from data collection to data mining and wrangling to the final database that is used for the web app. The connection to the website and its structure is shown, including how the user can search for SNPs and calculate summary statistics and generate plots.

Software Architecture

The database was built using `SQLite3` (Hipp, 2020) and integrated into the Website using the `Python` Standard (Van Rossum and Drake Jr, 1995) and `Flask` framework (Grinberg, 2018). Figure 2 shows how the main components were integrated into the web application. Red boxes show implementation and coding languages used, while the blue box shows the data that will be eventually seen on the user interface once a query has been made. The web application design is implemented using `HTML` templates and `CSS`/`JavaScript` stylesheets, which create the theme of the website. Information is posted on `HTML` templates in either the results or visualisation app route. Summary statistics and visualisation were implemented in `Python`, with "calculate" and "download" buttons provided in `HTML`. `CSS` stylesheets are connected to `HTML` templates by `href` links. Navigation bar, search bar, population and gene/SNP ID/position forms were created in `HTML` and `CSS`.

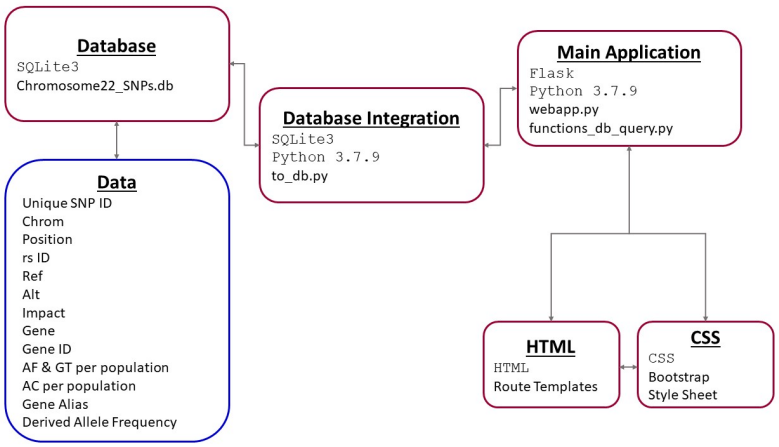

Figure 2: **Software Architecture.** The blue box shows the column names of the data in the SQL database table. Database connection to Flask and database query are done using `SQLite3` in `Python`. Calculating and plotting the statistics in the main application was done using `Python` scripts, as `HTML` and `CSS` are used to give the web pages professional looks.

Running the software

In order to deploy the website locally, download the *team_celine_webapp* directory from the GitHub page (see `https://github.com/graciaandr/bioinf_group_project`) and the database file in `.db` format from Google Drive using the following link: `https://drive.google.com/drive/folders/1g5-2Yr40tCxM30NLZQ0zmyzaBFjKPJIq?usp=sharing`

The user needs to ensure to download the database file into the `webapp` folder.

Install the required packages to the local machine by running:

```
$python -m pip install -upgrade pip
$pip install -r requirements.txt
```

Run the software on http://localhost:5000/ using:

```
$git clone
https://github.com/graciaandr/bioinf_group_project.git

$cd team_celine_webapp
$FLASK_APP=webapp.py
$flask run
```

Required packages:
Flask
Flask-Bootstrap
SQLite3
Scikit-allel
Pandas
Numpy
Matplotlib.pyplot
Seaborn
Pybase64 (base64)
Markupsafe

Flask and SQLite3

The software was built using `Flask` (Grinberg, 2018) because the tools it provides as a lightweight web framework allow beginners to learn setting up and running a web application. It is a micro-framework with little dependencies to external libraries, making it simple to use with clean and concise code. Furthermore, connecting `Flask` with the `SQL` database which was generated using `SQLite3` (Hipp, 2020) is straightforward using `SQL` query commands written in `Python`. User inputs from `HTML` forms can be easily processed and passed as parameters in data querying. Flask uses `jinja2` as its template engine, resulting in simplicity and efficiency in creating layout for the web pages. In addition, the feature of downloading data from the website as a text file was easily performed using a Flask route.

HTML/CSS

`HTML` coupled with `CSS` were used to deliver a presentable and professional website. `HTML` was used to format and display the web pages' contents, including displaying data in a table format and plot distribution images, while `CSS` allowed the addition of stylistic elements to the webpage. Bootstrap was also utilised to provide the page design template and navigation bar. `HTML` forms which consisted of text input, radio buttons, and checkboxes were generated to take user's input.

Data Collection

SNP Data

The SNPs used for creating the database were extracted from Phase 3 Chromosome 22 VCF file from the Ensembl FTP Server at http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data_collections/1000G_2504_high_coverage/working/20201028_3202_phased/CCDG_14151_B01_GRM_WGS_2020-08-05_chr22.filtered.shapeit2-duohmm-phased.vcf.gz.

Annotation Data

In order to link the gene regions the SNPs in the database lie in, annotated data from the Ensembl FTP server was used. The data can be found under the following link:
http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data_collections/1000G_2504_high_coverage/working/20201028_3202_raw_GT_with_annot/20201028_CCDG_14151_B01_GRM_WGS_2020-08-05_chr22.recalibrated_variants.annotated.txt

Population Metadata

Mapping the sample names to the super- and subpopulations was done by extracting the population samples from downloading the 3202 samples from the 100 genomes data portal (see https://www.internationalgenome.org/data-portal/data-collection/30x-grch38).

Gene Alias

As users are also permitted to look for SNPs by entering a gene alias into the search field, the annotated genes need to be linked to their correlating gene aliases. The `R` Bioconductor `org.Hs.eg.db` library contains these pieces of information and was used therefore. The data frame for connecting gene and gene aliases was stored as a CSV file for further data processing.

Derived Allele Frequency

To calculate derived and ancestral allele frequency using VCFTools, data was sourced from an annotated VCF file with ancestral allele (AA) in the info field. The file was collected from the 1000 Genome FTP server, which underwent downstream data processing (see http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/ALL.chr22.phase3_shapeit2_mvncall_integrated_v5b.20130502.genotypes.vcf.gz) .

Data Mining & Wrangling

Population Selection

The Phase 3 Chromosome 22 VCF file was filtered using `bcftools` (Danecek et al., 2021) to obtain 5 subpopulation specific VCF files. The chosen populations are:

- GBR - British (European)
- GIH - Gujarati (South Asian)
- LWK - Luhya (Africa)
- CDX - Dai Chinese (East Asian)
- MXL - Mexican (America)

Populations were selected due to the ability to cover wider genetic variation between populations and between SNPs also each population lives in different environments. We selected one representative subpopulation per superpopulation, and those with

a similar number of samples as GBR, which was the first population we chose. The VCF files were converted to CSVs and filtered by population using the following commands:

```
# filter VCF for samples of a selected population
# (saved in pop_samples.txt)
bcftools view -S pop_samples.txt
chr22.vcf
> filtered_for_pop.vcf

# remove "##" lines in vcf file and store it in CSV file
egrep -v "^##"  filtered_for_pop.vcf > filtered_for_pop.csv
```

After conversion to CSVs, only biallelic SNPs were retained using the Python data management library `pandas` (The pandas Development Team, 2020; Wes McKinney, 2010). The data sets were filtered for biallelic alleles for more manageable data processing when calculating reference, alternative allele counts and frequencies, as well as, genotype frequencies. The genotype was converted from numeric format to alphabet format in `to_db.py` in order to apply the `genotype()` and the `summary()` function from the `genetics` (Warnes et al., 2021) `R` package in the `calculate_frequencies.R` script (R Core Team, 2020), as these functions required the genotypes to be in letter format. The `genetics` package was more feasible than other packages which required a specific file format, e.g. `genepop`. The `summary()` function calculated allele counts and frequencies at SNP level and accepted only genotype data as the genotype object, which was generated by the `genotype()` function at SNP level.

Derived Allele Frequency Extraction

The ancestral annotated VCF file was required to calculate the derived and ancestral allele frequencies using `VCFtools` (Danecek et al., 2011). The derived and ancestral frequencies considered all populations in the 1000 genomes project to measure occurrence of non-ancestral/derived allele in the human population. Ancestral allele and chromosomes were extracted from the SNP VCF using `bcftools` and saved in a text file. Next, the text file was read as a data frame in `R` followed by extracting data with missing ancestral allele and insertion of chromosome number column as `VCFtools` required a text file in chromosome and number format. VCF filtered by positions using `VCFtools` and text file which contained the positions that should be excluded from the VCF file. These positions were removed from the VCF, as the `--derived` parameter in `VCFtools` command could not read VCF data which have missing ancestral alleles. Previous studies found derived/mutated alleles that arose after the divergence from the out-group (e.g. gorilla) by identifying ancestral alleles which are non-mutated alleles. This could imply allele frequencies for the alternative are the same for derived allele frequencies, as derived allele frequencies could be inferred when ancestral alleles are present (Koenig et al., 2019; Naji et al., 2021). Additionally, derived allele frequencies are usually much smaller than the ancestral allele frequencies (Gorlova et al., 2012). Derived parameter outputs the ancestral and derived allele frequencies using the following command:

```
vcftools -vcffile  data.vcf  -freq -derived -out derivedallele2

cat derivedallele2.frq | tr '\\t' ',' > derivedfreq.csv
```

Database Generation from CSV File

After data mining and wrangling, the data was put together into a table and saved as a CSV file. The SQL database `chromosome22_snps.db` was generated using `SQLite3` extension in `Python`. By executing the following commands, the `.db` file was automatically generated and a connection to the database was established.

```
connection = sqlite3.connect("chromosome22_snps.db")
cursor = connection.cursor()
```

A table was created in the database using `SQL CREATE TABLE` statement which required the column names, and the `unique_SNP_ID` column was assigned as the `PRIMARY KEY`. `insert_table()` function in `to_db.py` script was then used to populate the database table with the data in the CSV data file. Even though there was only one table containing our data, SQL database was still used because SQL database connection and query functions had already been written as multiple tables were generated originally in our database.

Database Connection and Query

HTML forms were utilised to capture user's input using radio buttons and input text box, which were then passed on to `Flask` using `request.form.get()` function. Query functions in `functions_db_query.py` were called using `search.db()` function to request and retrieve data from the database based on the search type (SNP ID, gene name/alias, genomic coordinates) the user chose and the search value the user typed in. The query functions established a connection to the SQL database using `sqlite3.connect` and `connection.cursor()` which then retrieved requested data from the database using `SQL SELECT` statement and `fetchall()`.

Data Visualisation

The summary statistics sliding window plot distributions were generated using `matplotlib.pylot` (alias `plt`) (Hunter, 2007), while the FST heatmap was generated using `seaborn` (Waskom, 2021). However, these plots could not be assigned into a variable to be passed on to the `HTML` to be displayed on the web page. Instead, `base64` – an encoding algorithm – was used to convert the image plots into a readable string that would be saved as a variable without any data loss (Matthieu, 2021).

```
img=io.BytesIO()
plt.savefig(img, format='png')
img.seek(0)
plot_url=base64.b64encode(img.getvalue()).decode()
image=Markup('<img␣src="{}">'.format(plot_url))
```

`Plot_url`, which contained the image data in strings, was then decoded using Markup to an `HTML` image tag `image` that could be returned to Flask and passed to `HTML` for the plot to be displayed on the web page.

Features of the Software

SNP Search

The search engine allows the user to search for SNPs based on three main features: gene name, gene alias, and genomic coordinates - so start and end positions in the chromosome. One singular position can also be entered instead of a range. The gene name looks for matches in the gene or gene alias attributes of the database.

Population Selection

From the provided populations, a user can select one or more, including all five of them, to calculate one or more of the given summary statistics.

Summary Statistics Selection

Shannon's Diversity

Shannon's diversity was calculated manually using Python by normalising the alternative allele frequency and calculating the natural logarithm of the normalised frequencies (see Equation . Although Shannon's diversity is affected by small samples, this would not impact the results as it is being applied to large data sets. Shannon's diversity illustrates diversity more effectively than using allele counts or richness. Also, it is commonly used for population genetics (Konopiński, 2020).

$$H_{Shannon} = -\sum_{i=1}^{N} p_i \cdot \ln(p_i) \; ; \; p_i \widehat{=} \text{ normalised alternative allele frequencies}$$

Expected Heterozygosity

A python function was built to calculate the expected heterozygosity per SNP using alternative and reference allele frequencies to estimate genetic diversity at locus level (Bepari et al., 2015; Harris and DeGiorgio, 2017). The following formula is applied per SNP:

$$H_{het} = 1 - \sum_{i=1}^{I} {p_i}^2$$

As I describes the number of alleles per SNP. Since only biallelic SNPs are in the database, I will always be 2 for each SNP , while p_i refers to the alterative and reference allele frequency per SNP.

Expected heterozygosity was chosen as a metric to compare the effect of urbanisation/agricultural society on genetic variation between populations and investigate physiological responses Rudan et al. (2008).

Tajima's D

Tajima's D was calculated for each SNP with the `tajima_d()` function from `scikit-allel` package (Miles et al., 2021), which used allele counts to work out dissimilarity between observed allele frequency in the VCF and expected frequency under neutrality (Ferretti et al., 2012). Chromosomal loci which underwent natural selection can be inferred by Tajima statistics when testing Wright Fisher model (Cadzow et al., 2014).

Hudson's FST

Hudson's FST method (extended by Bhatia et al.) in the `scikit-allel` package, which is the mean of FST values at population level defined by Weir and Hill, was recommended alongside the 'Ratio of Averages' approach when using multiple SNP data. The formula for calculating Hudson's FST according to Bhatia et al. is as follows:

$$FST_{Hudson} = \frac{FST_{pop1} + FST_{pop2}}{2}$$

The method 'Ratio of averages' was suggested as it does not affect FST estimates because they combine FST estimate (variances) by calculating the average of FST estimates in the numerator and denominator to provide a single FST value for the complete genome. Also, 'Ratio of averages' does not affect the combined FST value as FST ratio estimates corresponding to the numerator and denominator which are not biased/conflicting. Therefore, it provides the ability to merge to a single FST value for the complete genome, in comparison with the alternative 'Average of Ratios' approach.

Hudson is a suitable method for pairwise population analysis. Alternative methods such as Weir & Cockerham are influenced by differences in sample size, which leads to false positive signals as their estimates may be magnified (Bhatia et al., 2013).

Visualisation of Statistics

Sliding window created using python function which takes a `pandas` data frame and uses a fixed window size of rounded-up 10% of total queried SNPs. It is common for researchers to choose a random window size (Beissinger et al., 2015), however due to the limited time frame of the project in an extended project it would be possible to let the user decide what window size they want and make it robust against potential mistakes or misconceptions by the user (e.g. too large window sizes for a small query). To eventually decide which uniform window size to implement for the web application, different scenarios were tested: the complete data of approximately 930,000 SNPs was plotted with window sizes ranging from 1%, 10%, 25%, and 50% were plotted. The same window sizes were tested with smaller results, such as a data frame only containing around one hundred SNPs. The plots that showed the variability the most for the different scenarios, and what seemed reasonable to us, were the window sizes of around 10%. The window size will be rounded up, because we have to account for cases, where the data frame size is very small and 10% would be closer to 0 than 1, so this ensures that the smallest possible window size is 1 and not 0.

A line graph representing sliding windows was chosen as it is a suitable technique to show variation of statistical estimates (increase or decrease in estimates) throughout the chromosome or within a region, which could allow researchers to locate the position where a predicted gene partially overlaps and to possibly allow researchers to identify selection in a region. In comparison to overlapping windows, a distinct window size promotes statistical power as there is a reduction in sampling error and statistical test (Beissinger et al., 2014).

The y-axis shows statistical measurements for each window and the x-axis provides the median position value which corresponds to the SNP in the middle of the window as each window represents the average statistical value. To avoid gaps in the sliding window visualisation statistics were imputed with zeros. The FST

heatmap requires a data frame which contains the FST for any combination of populations for the `heatmap()` function from the `seaborn` package (Waskom, 2021). A heatmap was selected to visualise FST to highlight the genetic difference between populations (Fehren-Schmitz and Georges, 2016).

Download data

After the user selects their population and summary statistics of interest, the software will calculate the statistics, and it will put the statistics data frame into a text file. The user has the option to download the sliding window summary statistics data as a text file using the download button. If the user chooses more than one population, FST values for each population pair can also be downloaded as a text file.

Limitations

The database only stores biallelic human data from a single chromosome, and a limited number of subpopulations are included in the database due to data storage issues. Results produced from the website can't be applied to studies with x-linked chromosome diseases, as databases contain autosomal chromosomes. Also, unable to study genetic variation within and between other subpopulations which belong to either the same/different super populations.

When uniform window sizes are defined it could lead to regions where genetic events occur, such as recombination and linkage disequilibrium, being split when uniform window boundaries are defined. Shannon's diversity is affected by rare variants, therefore to compensate for error could use more than one compound indices (Morris et al., 2014).

Opportunities for future development

Using a non-uniform approach, could resolve the problem of breaking genetically significant regions. This approach is referred to as smoothing spline. The smoothing spline method fits cubic spline at SNP level statistical estimates (e.g. FST) followed by identifying inflection points along the model which defines window boundaries by ensuring peaks in the model are allocated to a window. Therefore, SNPs in a window can be inferred as having a relationship associated with genomic events. Cross validation determines smoothness of the spline before it's fitted and enables suitable window sizes (Beissinger et al., 2015). This technique is implemented in the `R` package `GenWin` (Beissinger, 2014).

The database should store more chromosomes to increase accuracy of the genetic analysis, which would consider SNP data which arises within the human genome to obtain a complete picture of genetic variation within and between subpopulations. Include more genetic statistics such as linkage disequilibrium (LD) which is the relationship between genes mapped at multiple chromosomal positions which is event by genetic events e.g. mutation and genetic drift (Slatkin, 2008). By finding LD value, researchers could use this to identify SNP associated with inherited diseases and measure the time an allele arises within the population.

Author's contributions

Implementation:

- Data Pre-processing for derived allele frequency (VCFtools) – Amanah
- Data Pre-processing (Bcftools) & Data Wrangling – Celine, Gracia
- Data Collection – everyone
- Web Deployment (Flask, HTML, CSS) – Celine
- Python to SQL database (SQLite3) – Celine
- Genotype & Allele Frequency Calculation – Amanah, Gracia
- FST statistic – Amanah
- Tajima's D and Shannon's diversity – Celine
- Sliding Window – Gracia

Documentation:

- Description of each contributed coding / implementation – everyone
- Justification of summary stats – Amanah
- Running of software – Celine
- Structure / Outline – Gracia
- Literature research / References – Amanah
- Transfer documentation into LaTeX– Gracia

Author details
[1]Queen Mary University of London, Mile End Road, E1 4NS London, UK. [2]AG Global Digital Health, Charité Universitätsmedizin Berlin, Charitéplatz 1, 10117 Berlin, Germany.

References

Beissinger, T.M.: GenWin: Spline Based Window Boundaries for Genomic Analyses. (2014). R package version 0.1. https://CRAN.R-project.org/package=GenWin

Beissinger, T.M., Hirsch, C.N., Vaillancourt, B., Deshpande, S., Barry, K., Buell, C.R., Kaeppler, S.M., Gianola, D., de Leon, N.: A Genome-Wide Scan for Evidence of Selection in a Maize Population Under Long-Term Artificial Selection for Ear Number. Genetics **196**(3), 829–840 (2014). doi:10.1534/genetics.113.160655

Beissinger, T.M., Rosa, G.J., Kaeppler, S.M., Gianola, D., de Leon, N.: Defining window-boundaries for genomic analyses using smoothing spline techniques. Genetics Selection Evolution **47**(1), 30 (2015). doi:10.1186/s12711-015-0105-9

Bepari, K.K., Malakar, A.K., Paul, P., Halder, B., Chakraborty, S.: Allele frequency for cystic fibrosis in indians vis-a/-vis global populations. Bioinformation **11**(7), 348–352 (2015). doi:10.6026/97320630011348. 26339151[pmid]

Bhatia, G., Patterson, N., Sankararaman, S., Price, A.L.: Estimating and interpreting fst: the impact of rare variants. Genome research **23**(9), 1514–1521 (2013). doi:10.1101/gr.154831.113. 23861382[pmid]

Cadzow, M., Boocock, J., Nguyen, H.T., Wilcox, P., Merriman, T.R., Black, M.A.: A bioinformatics workflow for detecting signatures of selection in genomic data. Frontiers in Genetics **5** (2014). doi:10.3389/fgene.2014.00293

Danecek, P., Auton, A., Abecasis, G., Albers, C.A., Banks, E., DePristo, M.A., Handsaker, R.E., Lunter, G., Marth, G.T., Sherry, S.T., McVean, G., Durbin, R., Group, .G.P.A.: The variant call format and VCFtools. Bioinformatics **27**(15), 2156–2158 (2011). doi:10.1093/bioinformatics/btr330

Danecek, P., Bonfield, J.K., Liddle, J., Marshall, J., Ohan, V., Pollard, M.O., Whitwham, A., Keane, T., McCarthy, S.A., Davies, R.M., Li, H.: Twelve years of SAMtools and BCFtools. GigaScience **10**(2) (2021). doi:10.1093/gigascience/giab008

Fehren-Schmitz, L., Georges, L.: Ancient dna reveals selection acting on genes associated with hypoxia response in pre-columbian peruvian highlanders in the last 8500 years. Scientific Reports **6**(1), 23485 (2016). doi:10.1038/srep23485

Ferretti, L., Raineri, E., Ramos-Onsins, S.: Neutrality tests for sequences with missing data. Genetics **191**(4), 1397–1401 (2012). doi:10.1534/genetics.112.139949. 22661328[pmid]

Gorlova, O.Y., Ying, J., Amos, C.I., Spitz, M.R., Peng, B., Gorlov, I.P.: Derived snp alleles are used more frequently than ancestral alleles as risk-associated variants in common human diseases. Journal of bioinformatics and computational biology **10**(2), 1241008–1241008 (2012). doi:10.1142/S0219720012410089. 22809343[pmid]

Grinberg, M.: Flask Web Development: Developing Web Applications with Python. " O'Reilly Media, Inc.", ??? (2018)

Harris, A.M., DeGiorgio, M.: An unbiased estimator of gene diversity with improved variance for samples containing related and inbred individuals of any ploidy. G3 (Bethesda, Md.) **7**(2), 671–691 (2017). doi:10.1534/g3.116.037168. 28040781[pmid]

Hipp, R.D.: SQLite (2020). https://www.sqlite.org/index.html

Hunter, J.D.: Matplotlib: A 2d graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007). doi:10.1109/MCSE.2007.55

Koenig, D., Hagmann, J., Li, R., Bemm, F., Slotte, T., Neuffer, B., Wright, S.I., Weigel, D.: Long-term balancing selection drives evolution of immunity genes in capsella. eLife **8**, 43606 (2019). doi:10.7554/eLife.43606

Konopiński, M.K.: Shannon diversity index: a call to replace the original Shannon's formula with unbiased estimator in the population genetics studies. PeerJ **8**, 9391–9391 (2020). doi:10.7717/peerj.9391. 32655992[pmid]

Matthieu, D.: Pybase64. (2021). Accessed: 22 February 2022

Miles, A., pyup.io bot, R., M., Ralph, P., Harding, N., Pisupati, R., Rae, S., Millar, T.: Cggh/scikit-allel: V1.3.3. doi:10.5281/zenodo.4759368. https://doi.org/10.5281/zenodo.4759368

Morris, E.K., Caruso, T., Buscot, F., Fischer, M., Hancock, C., Maier, T.S., Meiners, T., Müller, C., Obermaier, E., Prati, D., Socher, S.A., Sonnemann, I., Wäschke, N., Wubet, T., Wurst, S., Rillig, M.C.: Choosing and using diversity indices: insights for ecological applications from the german biodiversity exploratories. Ecology and Evolution **4**(18), 3514–3524 (2014). doi:10.1002/ece3.1155

Naji, M.M., Utsunomiya, Y.T., Sölkner, J., Rosen, B.D., Mészáros, G.: Investigation of ancestral alleles in the bovinae subfamily. BMC Genomics **22**(1), 108 (2021). doi:10.1186/s12864-021-07412-9

R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2020). R Foundation for Statistical Computing. https://www.R-project.org/

Rudan, I., Carothers, A.D., Polasek, O., Hayward, C., Vitart, V., Biloglav, Z., Kolcic, I., Zgaga, L., Ivankovic, D., Vorko-Jovic, A., Wilson, J.F., Weber, J.L., Hastie, N., Wright, A., Campbell, H.: Quantifying the increase in average human heterozygosity due to urbanisation. European Journal of Human Genetics **16**(9), 1097–1102 (2008). doi:10.1038/ejhg.2008.48

Slatkin, M.: Linkage disequilibrium — understanding the evolutionary past and mapping the medical future. Nature Reviews Genetics **9**(6), 477–485 (2008). doi:10.1038/nrg2361

The pandas Development Team: Pandas-dev/pandas: Pandas. doi:10.5281/zenodo.3509134. https://doi.org/10.5281/zenodo.3509134

Van Rossum, G., Drake Jr, F.L.: Python Reference Manual. Centrum voor Wiskunde en Informatica Amsterdam (1995)

Warnes, G., with contributions from Gregor Gorjanc, Leisch, F., Man., M.: Genetics: Population Genetics. (2021). R package version 1.3.8.1.3. https://CRAN.R-project.org/package=genetics

Waskom, M.L.: seaborn: statistical data visualization. Journal of Open Source Software **6**(60), 3021 (2021). doi:10.21105/joss.03021

Wes McKinney: Data Structures for Statistical Computing in Python. In: Stéfan van der Walt, Jarrod Millman (eds.) Proceedings of the 9th Python in Science Conference, pp. 56–61 (2010). doi:10.25080/Majora-92bf1922-00a