Software Development Group Project

MSc Bioinformatics 2022/2023 QMUL

Team Vivaldi

Type-1 Diabetes SNP Search Tool

This software was developed for the Software Development Group Project module of the MSc Bioinformatics 2022/2023 at Queen Mary University of London. The group members are Khadija, Marta, Jiang, Thuvarahgan and Tobi.

Repository link: https://github.com/khadijapaderwala/team-vivaldi.git

Table of Contents

Introduction

Genome wide association studies (GWAS) have been able to identify single nucleotide polymorphisms (SNPs) that are linked to Type 1 Diabetes (T1D). Databases such as SNPnexus can provide information about SNPs such as functional information, functional impact or clinical relevance. The aim of this project was to produce a functioning web-based software tool that allows users to search and explore genetic variants that have been associated with high susceptibility of Type 1 Diabetes. The tool also retrieves genomic information and integrates it with population data and functional information. We have called it 'T1D SNP Search Tool.'

The website allows users to search for genetic variants by single nucleotide polymorphism (SNP) name (rs ID), genomic position or gene name. For each query, the web page returns information on SNP name (rs ID), genomic position, p-value from association tests, mapped genes, allele frequency in three different human populations (British GBR, Nigerian ESN, Japanese JPT), one measure of functional impact and one functional term associated with each mapped gene. When users search the website for SNPs by genomic region range, the webpage returns a Manhattan plot of all the p-values returned from the search. The webpage has an option to select SNPs, calculate linkage disequilibrium (LD) of those SNPs, and return plots for those LD measures. We aim for this tool to be used to interpret the mapping of the genetic basis of T1D.

It is important to note, the software functionalities described above are applied to only genetic variants of chromosome 6. However, there is also some additional data available on the website for other chromosomes.

Getting Started

To get started, you first need to create a virtual environment. To do this open terminal on your local computer and then follow the steps below:

1- Install virtualenv in global python: `pip install virtualenv`
2- Create virtual environment: `python -m venv venv`
   The second venv in the code is what the virtual environment will be called, you can choose to name it something else if you please.
3- Source the environment:
   For Mac users: `source venv/bin/activate`
   For Windows users: `venv\Scripts\activate`

Once your virtual environment has been created you can install the dependencies used for this project by uploading the 'requirements.txt' file in the newly created virtual environment. Open terminal again and follow the step below:

4- Install project dependencies: `pip install -r requirements.txt`

Download the following files and folders into the virtual environment from GitHub repository (https://github.com/khadijapaderwala/team-vivaldi.git):

- 'main.py'
- 'src' folder containing- LD_calculation_function.py, LD_outputs.py, LDheatmap.py, manhattan_plot.py
- 'templates' folder containing- index.html, 404.html, LD.html, SNP.html, Gene.html, Region.html, region_single.html
- 'Database' folder containing- Database.db, british_chr6.db, japanese_chr6.db, nigerian_chr6.db
- Create an empty folder named 'static'

You can then run the website within the virtual environment by running the following code in terminal using this command: `python main.py`

Software Architecture

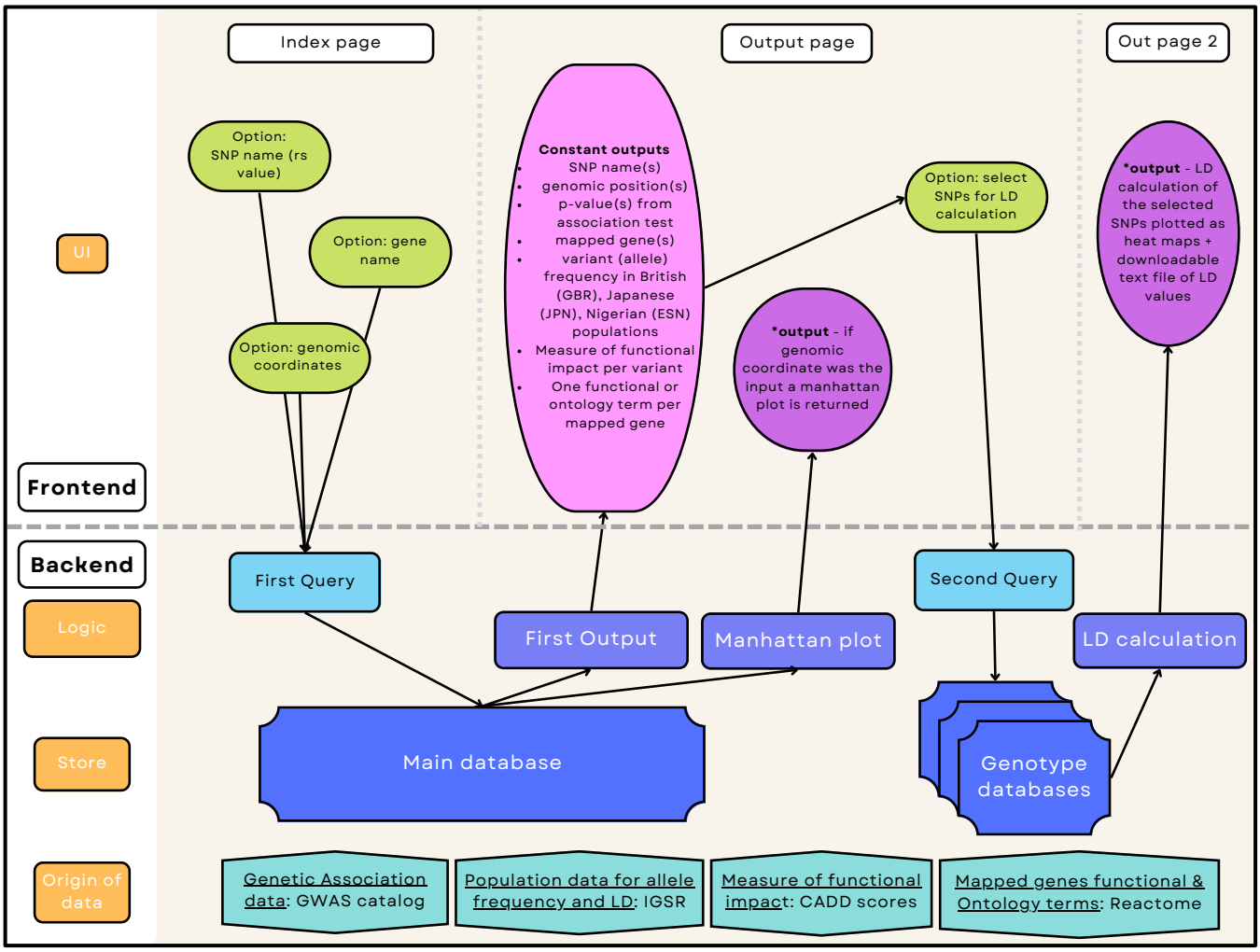


Figure 1. Diagram of the Software architecture. Once the website is up and running, the user starts on the index page. It has the option to input three types of data: SNP rs ID, gene name, or genomic coordinates. Next, the user will always be given the results listed in the 'constant output' oval. If the input was a genomic coordinate, the software will return a Manhattan plot. On the constant output page, users will have the option to calculate LD for any SNP they like. This will output 6 heatmaps for each LD measure, for each population (r2, D', for the British, Nigerian, Japanese populations). They will also have the option to save the LD calculation results in a .txt file.

The initial page the user will land on is the index page, from this page, users will be able to provide an input. The input can be SNP name, gene name or genomic coordinates. If the user enters something that is recognised in the database. The first query is directed to the general SQL database which contains information about SNP name, genomic positions(s), p-values(s) from association test, mapped gene(s), variant (allele) frequency in at least 2 human populations, at least 1 measure of functional impact per variant, at least 1 ontology term per mapped gene. The SQL database was populated with this information from the following online databases: Genetic Association Data-GWAS catalogue, IGSR, CADD and Reactome. From SQL, the user query is then outputted to another webpage. If the user query was a range of genomic coordinates, a Manhattan plot will be a part of this constant output page.

On the constant output page, there will be an option for users to input a second query, they can input multiple SNP IDs to calculate linkage disequilibrium between pairs of SNPs. The second query is

processed, and the haplotype data required by the LD calculations is obtained from three genotype databases, one for each population. When LD is calculated, it is returned on a second output page along with a downloadable text file of LD values and heatmaps for the LD values.

Data Wrangling

The data included in the databases was obtained from various sources. We have explained how we collected and used this data below. Everything used to create the database can be found in the development folder.

Genome Wide Association study (GWAS) data

We obtained T1D GWAS data from the NHGRI-EBI Catalog of human genome-wide association studies to investigate the genetic basis of Type 1 Diabetes (Sollies et al., 2022). The NHGRI-EBI Catalog is a database that includes information on published genome-wide association studies from various populations worldwide. The results of this search included information on the genetic variants, their frequencies, and effect sizes on the risk of type 1 diabetes from various populations.

By analysing this GWAS data, we can gain insights into the genetic basis of Type 1 diabetes across different populations and identify potential genetic targets for the development of new treatments or prevention strategies.

SNP Measures of Functional Impact

As a measure of functional impact, we chose CADD scores for the SNPs. CADD (Combined Annotation Dependent Depletion) is a score that integrates functional annotations, conservation, and other features into a machine-learning algorithm to predict the deleteriousness of genetic variants (Rentzsch et al., 2021). CADD Phred is a normalized and transformed version of the CADD score that ranges from 0 to 99, where higher scores indicate a greater deleteriousness of the genetic variant, thus scores are values expressing the rank in order of magnitude rather than the precise value itself. They can then be used to prioritize potential functional variants for further investigation. We obtained these scores by querying SNPnexus using the list of rs IDs for chromosome 6 obtained from the GWAS dataset (Oscanoa et al., 2020). SNPnexus obtains the CADD scores from Ensembl. The resulting text file contained the rs ID, chromosome position, raw CADD score, and CADD-PHRED score. The CADD-PHRED scores were added to the main database using the corresponding rs IDs.

Gene Functional terms

Gene functional terms are terms that describe the biological process, cellular component, or molecular function. A database that can be used to obtain gene and SNP functional data is SNPnexus (Oscanoa et al., 2020), which integrates data from multiple sources, including the Reactome pathway database (Gillespie et al., 2021). Reactome is an open-source pathway database. There are many pathways present in Reactome such as disease, signalling and intermediary metabolism.

This data was used to gather the mapped genes and their corresponding function. While most of the data is pathway related, some mapped genes also have more detailed information such as molecular function. Any missing functional data for the mapped genes was gathered from the GeneCard database (Safran et al., 2022). Primarily the functional terms for the mapped gene were from the 'parent' column of the SNPnexus Reactome dataset. However, as some of the terms were too general such as 'Immune', 'Disease', 'Signal Transduction' and 'Metabolism', they were supplemented further with more specific functional terms from the GeneCard database.

SNPnexus also outputted an ensembl file, which provides ESN identifiers, which was used to map the genes to the SNPs (Cunningham et al., 2022). This is used to populate the join table between Gene and SNP.

Gene functional data allows the user to explore the functional implications of genetic variants such as their effects on protein function and interactions, as well as their potential roles in disease. Additionally, it allows the user to explore the data in the context of specific pathways, which can help researchers to identify potential mechanisms underlying the genetic associations with Type 1 Diabetes and prioritize genetic variants for further investigation.

Population data

The 1000 genomes project has successfully established detailed information regarding human genetic variation. This is possible as the individual haplotypes of many participants have been mapped to a reference human genome. The 1000 genomes project data is available at the International Genome Sample Resource (IGSR) website, this is where we obtained all our population data from (The 1000 Genomes Project Consortium, 2015). The IGSR website is an ideal data source as it contains the genetic data from the 1000 genomes project for all chromosomes of the genome so that later we can expand our web browser to the entire genome.

The data collection was filtered to 1000 genomes on GRCh38 data. We have selected three populations to calculate the population allele frequencies and linkage disequilibrium however you can choose any suitable populations and how many populations based off your preferences. The three populations we have selected are:

- British in England and Scotland (GBR – 100 sample IDs)
- Esan in Nigeria (ESN – 100 sample IDs)
- Japanese in Tokyo, Japan (JPT – 105 sample IDs)

We have selected the GBR population as a reference population as we believe most potential users of our website will be working from the UK, thus working with population data from the UK. We selected ESN and JPT populations as both are geographically distant from the GBR population and believe geographical location may shed light on genetic differences in T1D. Eventually, our aim is to include all 1000 genomes populations in our tool.

We have initially used data from chromosome 6 only to test our software. We downloaded the vcf.gz file corresponding to variants on chromosome 6. In addition, we downloaded the list of sample IDs for every population we used in separate files. The sample IDs are also available at the IGSR website. In the future, we aim to expand the population data to the entire genome.

Using tools like bcftools and SNPnexus we filtered the VCF file and retained a more specific VCF file that consisted of the genotypes of the desired population only and SNPs linked to T1D. One of the

files from SNP nexus was called 'gen_coords.vcf', which included the genetic coordinates for all the rs IDs inputted in the query. Based on these coordinates, we were able to fill the empty rs ID column in the population VCF files. These VCF can be found in 'development/LD/< population >_filled_in_IDs.vcf'. The README.txt in the development/LD/ directory explains all the steps and files used to fill the columns. We retained only the rs IDs linked to type 1 diabetes using the T1D GWAS SNP data we have downloaded from the NHGRI-EBI GWAS catalog available at: https://www.ebi.ac.uk/gwas/home. The file we used is called: ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data_collections/1000_genomes_project/release/20190312_biallelic_SNV_and_INDEL/ALL.chr6.shapeit2_integrated_snvindels_v2a_27022019.GRCh38.phased.vcf.gz

We created a python function called 'Allele_Frequency' which took the filtered VCF file as input and generated a text file of allele frequencies for every SNP in the population. The two inputs required from the user are the filtered VCF file and the name of the output text file to be saved. We have also commented on the source code to help the user understand the function better whilst using it. Please refer to the ***Allele_Frequency.py*** file in the GitHub repository to use the function. A text file with the allele frequencies of each SNP for the population is returned.

The text files for all the desired populations' (GBR, ESN and JPT) allele frequencies were taken and merged into one csv file. Please refer to the Pop_Allele_Freq.csv in the GitHub repository to see the raw csv file. This csv file has been incorporated into the SQL database for our web browser so that when a user searches a request, the corresponding allele frequency column will be called and be returned as an output.

Haplotype data for Linkage Disequilibrium Calculations

The haplotype data required for the LD calculations was obtained from the population VCF files mentioned above. The population genotypes for each rs ID were extracted and separate databases were created, one for each population. The databases can be found in the 'Database' directory, and are named the following:

- british_chr6.db
- nigerian_chr6.db
- japanese_chr6.db

The databases are currently set up with 62 tables each, with one table representing one rs ID, and the genotype data in the rows below. There are 100 genotypes for the British population, 100 genotypes for the Nigerian population, and 105 genotypes for the Japanese population. Presently, these databases only contain the genotype information of the chromosome 6 rs IDs related to T1D. The genotypes of the populations are phased, therefore enabling the LD calculations as the calculations require haplotype information. A haplotype is a combination of alleles at multiple loci that are transmitted together on the same chromosome. In the future, we aim to have all chromosome data in our databases for each population.

Front-end

Note

The main code was written in main.py, there were functions imported into main.py from other written code in the src folder in the virtual environment. These functions were 'manhattan_plot' from src.manhattan_plot and 'LD', 'write_table_to_file', 'heatmap' from src.LD_outputs. Also, the SQL database used was Database.db, this database contained all information returned from the first query. This was rs ID, location, reference/alternate allele, population allele frequency from GBR/JPT/ESN populations, CADD, P-values, mapped genes and functional information. The second query which is linkage disequilibrium calculation retrieves its output from the british_chr6.db, japanese_chr6.db and nigerian_chr6.db SQL databases.

Flask

Flask was the web framework used to develop this website (Grinberg, 2018). Python Flask is a lightweight and flexible web framework used for building web applications in Python. Flask provides basic functionality such as request handling, routing, and response generation.

The index page of the website has a title and a small description for users to understand the purpose of the website. There is a small note telling users how to format their input in the search bar. This is done because there are 4 different possible webpage routes (not including the index page itself), the input can take them to. When the user inputs an rs ID, it takes them to the following route: '/SNPs/<id>'. When the user inputs a single genomic position, it takes them to the following route: '/Region/<chr_n>/<chr_p1>'. When the user inputs a genomic region range it takes them to the following route: '/Region/<chr_n>/<chr_p1>-<chr_p2>'. When a user inputs a gene name, it takes them to the following route: '/Gene/<id>'. The route depends on what the user is searching. This is made possible using regular expression and recognising patterns in the query inputted.

To recognise what the user is searching, the software checks to see if the first two letters of the string is 'rs'. If it is, then the page redirects to the SNPs URL, if not the software checks to see if the first three letters of the string is 'chr'. If it is then, it can redirect to either the URL for Region_range or Region_single. To distinguish whether the user is searching for a single genomic position or a region range, a nested loop is used. If the user enters 'chr' as the first three letters and a dash symbol (-) in the same string, then it will be treated as a search for genomic range. Otherwise, if the user query contains 'chr' as the first three letters, but the string contains no dash symbol (-), it will be treated as a search for a single genomic position. Lastly, if the user has entered an input without 'rs' as the first two characters or 'chr' as the first three characters, it will assume a gene name is being searched. For this reason, it is important for the user to have a note to format their query correctly. For example, if they search for SNP using 'RS' instead of 'rs', then the software will return an error message. Another example is if the user searches for genomic position by 'chromosome', instead of 'chr', then an error message will be presented. To attempt to handle formatting issues, not only is there a description above the search bar, but the error message returned also reminds users how to format the query correctly.

The reason regular expression pattern recognition was used was because the alternative was to have multiple search bars on the webpage, each taking you to either the SNP, genomic position, genomic region or gene route. However, this made the webpage look too cluttered, therefore we opted for one search bar filtering by regular expression. A future improvement to this could be adding more

elaborate and advanced pattern recognition options which could, for example, allow users to type the rs ID in uppercase letters but still identify it as a SNP search.

The search bar on the index page was made using the StringField and SubmitField class from wtforms. StringField allows users to enter their input and SubmitField sends the request. DataRequired from wtforms.validators means users must provide an input into the StringField instance before clicking submit, if it is left blank the software will prompt the user to fill out the field. NameForm is the class definition which inherits the FlaskForm function. Instances of StringField and SubmitField are contained in the NameForm class. An instance of NameForm class is later created and stored in the variable 'form'. This is then used to search the Database.db SQL database for the user input and return the output page.

When the user searches with rs ID, gene name, chromosome position or chromosome range, and the input is found in the database. The output returned is:

- Rs id
- Location
- Reference allele
- Alternate allele
- GBR reference allele frequency (British population)
- GBR alternate allele frequency (British population)
- JPT reference allele frequency (Japanese population)
- JPT alternate allele frequency (Japanese population)
- ESN reference allele frequency (Nigerian population)
- ESN alternate allele frequency (Nigerian population)
- CADD scores
- P-value
- Mapped Genes
- Functional information

Information about rs ID, location, reference allele, alternate allele, GBR/JPT/ESN reference and alternate allele frequency, CADD scores are found in Database.db SQL database in table 'SNP', p-values are found in table 'P_value', mapped genes are found in table Gene_SNP and functional information is found in table 'Gene'. The tables are joined together in main.py using JOIN SQL statements so that although the information is found in different tables, it can be returned in one output.

When a SNP is searched by the user, the tables are searched by rs ID. The output is rendered by the SNP.html template. When user searches by single chromosome position (chr[int]:[int]), the tables are searched by chromosome number and chromosome position and the output is rendered by the region_single.html template. When the user searches by a genomic region range (chr[int]:[int]-[int]), the tables are searched by chromosome number and all the rows between chromosome position 1 (before the dash symbol in the user input) and chromosome position 2 (after the dash symbol in the user input), results are rendered by the Region.html template. Finally, when the tables are searched by gene (any valid user input that does not begin with 'rs' or 'chr'), the tables are searched by gene name and the output is returned by the Gene.html template.

On each output page, there is a search bar so that users can input rs IDs separated by a comma and get a linkage disequilibrium calculation of those rs ids. From creating_heatmap.py in the src folder,

the functions ‘LD’ and ‘write_table_to_file’ were used to calculate LD and the function ‘heatmap’ was used to create a heatmap of LD.

The specification required LD calculation when only chromosome region was given, and the user should be able to select the outputted SNPs to calculate LD. However, we have incorporated an LD search bar on every output page so that the user has better access to the LD function. We also opted for a search bar rather than a checkbox option so that the user can input any rs ID and providing that it exists in the databases, it will give an output. Whereas with a checkbox option, users can only select SNPs that have outputted on the webpage, which is much more limiting.

To create the search bar for LD calculation a new Flaskform class was created named LDForm. Similarly, to the NameForm class used for the index page, this class contains StringField and SubmitField classes within it. The StringField instance gives a small description to users to provide at least two rs IDs for LD calculation separated by a comma. If the input is valid once the user clicks the calculate button, it searches for the rs IDs in the british_chr6.db, japanese_chr6.db and nigerian_chr6.db. Each database contains the haplotype of every individual used in the population study, this is then used to calculate LD.

When users search by a genomic region range, it is required to have a Manhattan plot with all the p-values of the outputted SNPs plotted on them. The chromosome number being searched is the x-axis and the p-value is plotted along the y-axis. The manahttan_plot function is imported into main.py from another python file containing the function called manhattan_plot.py in the src folder.

When the user formats their query in the correct manner, but the rs id/gene name/position does not exist in the SQL database. The 404.html template will be rendered to tell users the query is not in the database, and they will have the option to return to the search page. In the case where the user is not appropriately formatting their query, then they will receive an error message on the index page with a reminder on how to format correctly.

Templates

When the user puts in an input on the index page, one of five templates are shown. If search is by SNP, then the SNP.html template is rendered, if the search is by single chromosome position, then the single_region.html template is rendered, if the search is by a chromosome region range, then the Region.html template is rendered, if the search is by gene, then the Gene.html page is rendered. Finally, if the user input is not found in the database, then the 404.html template is rendered.

In main.py, these templates are rendered by the render template function from flask. The templates are all in a template folder because flask by default searches for templates in a template folder.

In main.py there is a snippet of code: Bootstrap(app). This snippet of code is used to activate the bootstrap framework. Doing this is allows developers to use bootstrap components and designs for a visually appealing webpage. Bootstrap is then referenced in the html templates (excluding LD.html).

The bootstrap inheritance statement in the html template (`{% extends 'bootstrap/base.html' %}`) allows the template to inherit the structure and content of the parent (base.html) template which defines the structure of a html page. The `{% import "bootstrap/wtf.html" as wtf %}` statement allows macros to be imported from wtf.html which allows the rendering of forms with bootstrap styles.

In the html template the function ‘super()’ calls the parent template (base.html) of the styles blocks, so any style in the parent template is inherited by the child template (SNP.html/Region.html/Gene.html/region_single.html). The child template specifically sets the background colour to light blue which will be rendered, in addition to any other style existing in the parent template.

In the html template SNP.html/Region.html/Gene.html/region_single.html there is also the following snippet of code: {{ wtf.quick_form(form) }}. This line of code refers to variable ‘form’ in the main.py code, which is a search bar for LD calculation. By adding this line in the html template, the output page will render a search bar for users to put input rs IDs for LD calculation.

The rest of the html template contains the names of the columns as they appear on the webpage and the row index the column name correlates with. The row index block is in a for loop, to give more than one output if multiple matches are found in the database for a single query.

Databases

Main Database

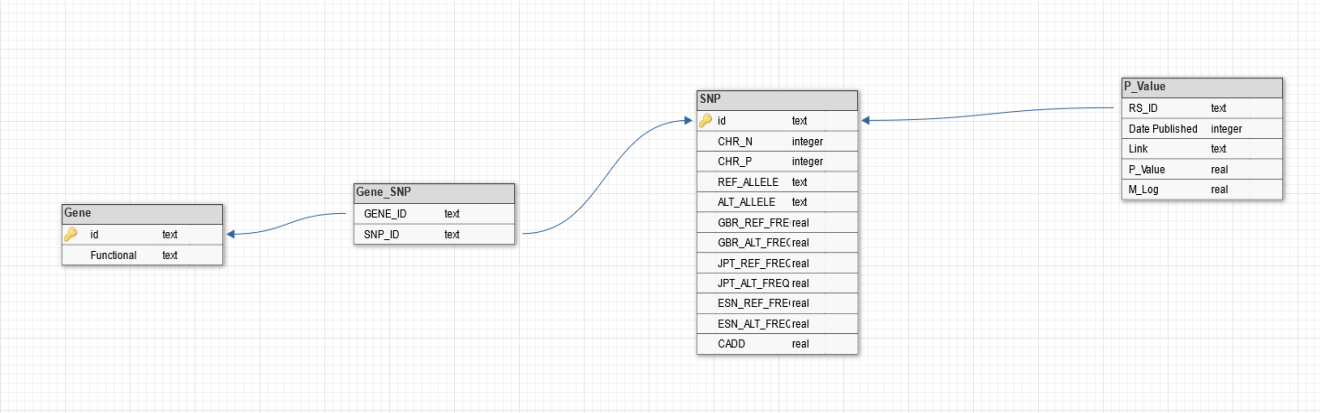


Figure 2. Current schema of the main database. There are 4 tables, and rs ID is the primary key.

In the above figure shows the schema of the database. The id field of the SNP table is where all the rs IDs are stored and serves as the main primary key. The unique nature of the SNP names is why it serves as a primary key. All the fields on the SNP table have a one-to-one relationship with the SNP name so they were put in the same table. There can be multiple p-values for a single SNP name so another table for p-value was constructed to better illustrate and search through this one-to-many relationship. As the relationship between genes and rs IDs is a many-to-many relationship, a join table (‘Gene_SNP’) was created.

The database can be generated using the ‘datasets’ folder which is in the ‘development’ folder with the ‘Database_Creation.py’ script and ‘Database.sql’ (in the ‘development/Data_Functions’ folder) which connects to various dataset files wrangled as mentioned above using the relative file path described in the ‘Database_Creation.py’ script and populates the ‘Database.db’ file. ‘Pop_Allele_Freq.csv’ was used to populate the SNP’s table id field along with the reference and alternative allele and the three populations of interest with their allele frequencies. From the GWAS dataset, the rs IDs with their respective chromosome number and position were imported into the SNP table as well as all the values in the ‘P_value’ table. The ‘Gene_SNP’ table was populated with

data from ensembl found in the datasets folder called ‘ensembl.txt’ using the ‘Symbol’ and ‘Variation ID’ column. The ‘Gene’ table was populated with data from ‘Gene_Term_Updated.txt’. This file is an amalgamation of various sources (SNP nexus’s pathway.txt and GeneCard searches for entries that are lacking) cleaned and curated for the database to fit for this table. CADD scores were updated with the ‘CADD_C6.csv’ where based on the value found in ‘id’ of the SNP table, it will update the CADD field for that particular SNP name using the ‘SNP’ and ‘PHRED’ columns in ‘CADD_C6.csv’.

The app uses sqlite for the database. Sqlite is a free open-source database engine and is embedded within the application. It is serverless and portable, making it easy to implement. Datatypes of sqlite is limited to *real, integer, text, blob* and *null* (Hipp, 2020). As shown on the figure above, most of the columns uses a *text* datatype apart from fields that have numeric values. The allele frequencies for each population, CADD scores and p-values uses the *real* datatype as it is stored in decimal places. Chromosome numbers and positions along with the date published associated uses the *integer* datatype as the values for these fields are whole numbers.

Population Databases

The three population databases (‘british_chr6.db’, ‘nigerian_chr6.db’, ‘japanese_chr6.db’) use the same technology, Sqlite, as the main database. These are much smaller and are not connected to the main database. They are also not connected to each other. The databases are structured in the same way, with one table per rs ID. Each table contains the genotype information for that rs ID for that population. These databases are only in use when a user decided to input rs IDs in the LD search bar.

Calculations and Plots

Manhattan Plot

We have incorporated a Manhattan plot into our web browser as a visual aid for the comparison of p-values of SNPs located close to each other when the user requests a genomic coordinate range. The Manhattan plot uses data from the T1D GWAS SNP data we have obtained. The GWAS data already contains the relevant values for SNPs to create the Manhattan plot such as rs ID, chromosome number, chromosome position and p-value. We have created a python function that takes only the rows of data from the original GWAS database, for SNPs that are a direct output of the user search request. This SNP data is saved into a new data frame and sorted by chromosome position. The –log10 of the p-values can then be calculated and then a plot is generated using the matplotlib module in python. This Manhattan plot is saved as a png file in our web browser’s static folder. Please refer to the ***manhattan_plot.py*** file in our GitHub repository. Therefore, when a user searches by genomic coordinates on our web browser and multiple SNPs are returned, a Manhattan plot of the SNP p-values is displayed at the top.

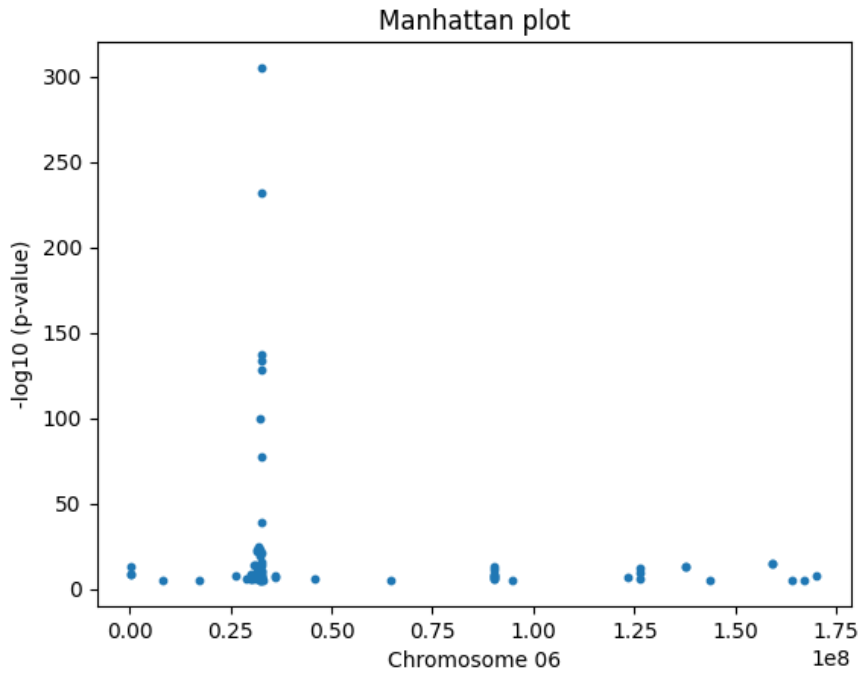


Figure 3. Example of a Manhattan plot for all SNPs in chromosome 6. Chromosome positions on the x-axis, and -log10 p-values on the y-axis.

Linkage Disequilibrium Calculations

The linkage disequilibrium (LD) is the non-random association between alleles at different loci. When alleles occur together more often that would be expected by chance, they are said to be in linkage disequilibrium. There are various measurements for LD: D, r^2, and D' (*Equations 1, 2, 3*). D is the coefficient of LD and is not often used as a standalone measure of LD because the value depends on what allele is being chosen to calculate the values of PAB, PA and PB. Therefore D', the normalised coefficient of D, and r^2, the squared correlation coefficient, are considered more informational and often used in literature.

The values of D' range between -1 and +1. The ends of the range mean there are no mandatory recombinants between markers. A higher D' value means two markers are better surrogates for each other. Unfortunately, D' estimates are inflated in small samples and when one allele is rare. A D' value of ±1 implies at least one of the observed haplotypes was not observed.

The values of r^2 range between 0 and 1. The r^2 value of 1 means two markers provide identical information, and a value of 0 mean they are in perfect equilibrium.

Equation 1. Formula for coefficient of linkage disequilibrium (Lewontin and Kojima, 1960); D.

$$D = P_{AB} - P_A * P_B$$

Equation 2. Formula for normalised coefficient of LD; D' (Hill et al., 1968).

$$D' = \frac{D}{D_{\max}}$$

where

$$D_{\max} = \begin{cases} \max\{-p_A p_B,\ -(1-p_A)(1-p_B)\} & \text{when } D < 0 \\ \min\{p_A(1-p_B),\ (1-p_A)p_B\} & \text{when } D > 0 \end{cases}$$

Equation 3. Formula for the squared correlation coefficient between two indicator variables; r^2 (Hill et al., 1968).

$$r^2 = \frac{D^2}{p_A(1-p_A)p_B(1-p_B)}$$

All the scripts responsible for these calculations are found in the 'src' folder. The python file 'LD_calculation_function.py' contains a function that obtains the genotypes for a pair of rs IDs by connecting to the databases of the population genotypes. The function then counts the haplotypes, calculates the frequencies of PA, PB, and PAB (required for the formulas), then calculates the coefficient of LD (D), the normalised coefficient of LD (D'), and the square of the correlation coefficient (r^2). The output of the file is a dictionary that states the pair of inputted rs IDs, the haplotype counts, the allele frequencies of those haplotypes which are used in the equations, D, D', and r^2 (for each population).

The frequencies of PA, PB, and PAB are calculated based on the haplotypes found in the same chromosome of two rs IDs. A basic example calculation is shown in Figure 4. In the example, the genotypes are for two fictional rs IDs are highlighted with the blue rectangles and the haplotypes are highlighted with red rectangles. The script takes note of how many haplotypes it encounters. A '00' haplotype means on one chromosome, that individual has a reference allele at rs ID1 location and a reference allele at rs ID2 location. A '01' haplotype means on one chromosome, that individual has a reference allele at rs ID1 location and an alternate allele at rs ID2 location. A '10' haplotype means on one chromosome, that individual has an alternate allele at rs ID1 location and a reference allele at rs ID2 location. A '11' haplotype means on one chromosome, that individual has an alternate allele at rs ID1 location and an alternate allele at rs ID2 location.

The PA frequency is the sum of the counts of the '00' and '01' haplotypes, divided by the total number of haplotypes. The PB frequency is the sum of the counts of the '00' and '10' haplotypes divided by the total number of haplotypes. The PAB frequency is the number of '00' haplotype counts divided by the total number of haplotypes.

Next, the code uses the PA, PB, and PAB frequencies to calculate the coefficient of LD, D. The same values are used to calculate the normalised coefficient, D', and the squared correlation coefficient, r2.

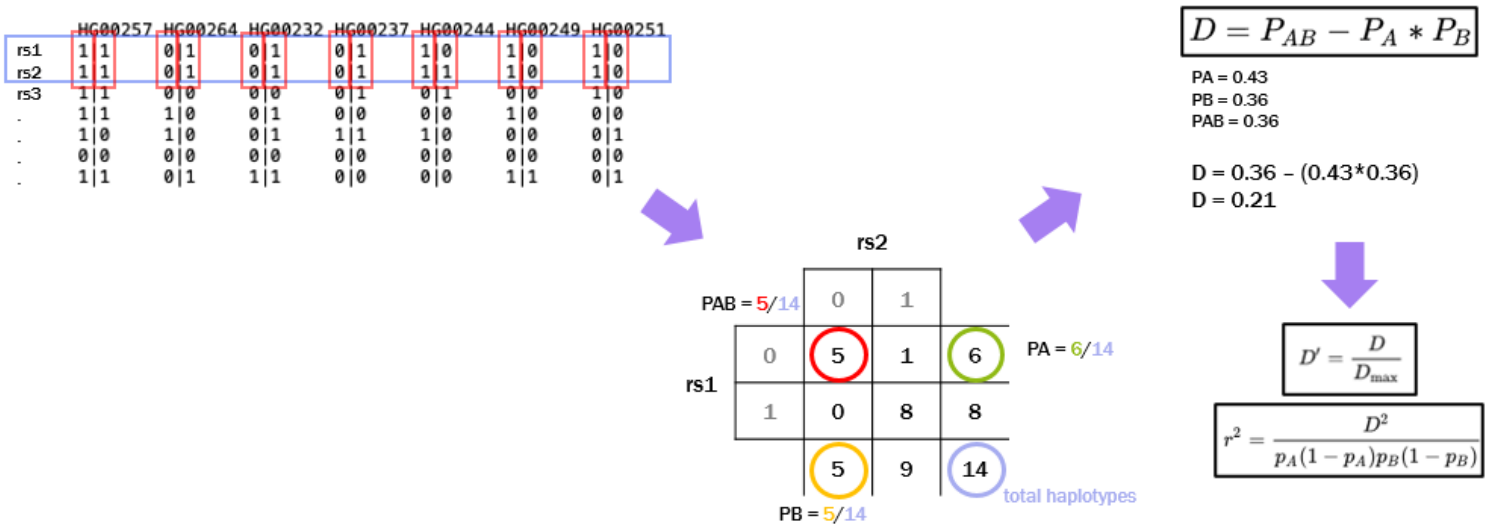
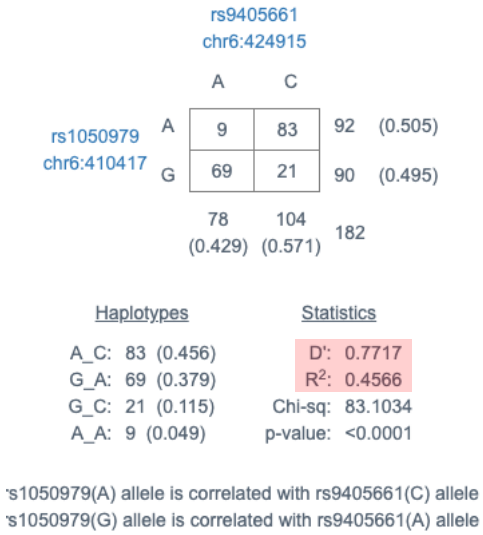


Figure 4. *A basic example of how the PA, PB, and PAB frequencies are calculated by the code, for the LD formulas. PA is the number of haplotypes '00' and '01', divided by the total haplotypes count. PB is the number of haplotypes '00' and '10', divided by the total haplotypes count. PAB is the number of haplotypes '00' divided by the total.*

During development, the results were compared to NCBI's LDpair Tool. While our results were not exactly the same, they were comparable. The slight differences come down to a slightly different number of individuals in our populations, even though both our software and theirs uses 1000 genome population data. https://ldlink.nci.nih.gov/?tab=ldpair An example of the comparison can be seen in figure 5.



rs ID_1	rs ID_2	British haplotypes	British pA	British pB	British pAB	British D	British r2	British Dprime
rs1050979	rs9405661	{'00': 92, '01': 9, '10': 22, '11': 77}	0.505	0.57	0.46	0.17215	0.48369786	0.79276997

Figure 5. Screenshot of the results on the LDpair webtool for rs1050979 and rs9405661 for the British GBR population (left) and a snippet of our text file of the LD calculation results (right). The red highlights show the r2 and D' values. It is important to note that in their calculations for s9405661, they consider C as the reference allele of C and A as the alternate, while we considered the opposite in the calculations.

Linkage Disequilibrium Calculation Outputs

All the code for the LD calculations, LD heatmap and LD results text file can be sound in the 'src' directory. The files are the following:

- LDheatmap.py
- LD_calculation_function.py (file contents mentioned in section above)
- LD_outputs.py

The technologies required for these three files are Numpy, Matplotlib and Itertools for creating the heatmap, and Sqlite for accessing the population databases. For the LD_outputs.py file, the requirements are Matplotlib, CSV, and the functions described in LDheatmap.py and LD_calculation_function.py. The main file, main.py calls to the functions in LD_outputs.py (Figure 6).

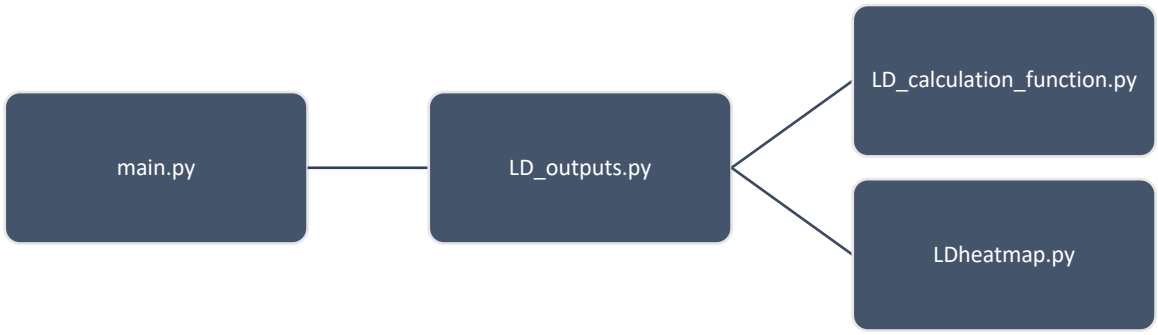


Figure 6. Diagram of how the main script links to the LD scripts. Main.py only calls to the functions in LD_outputs.py.

The results of the LD calculations depend on the user's input. The more rs IDs the user decides to calculate LD for, the bigger the heatmaps and bigger the text file. Due to the functions being called on demand in the user interface, the calculations and plots are created very quickly, in comparison to using an API for pre-existing tool.

The file 'LDheatmap.py' contains a function called 'LDheatmap' that intakes a list of values (e.g. British D' values), a list of rs IDs, and a title for that heatmap. The output is a heatmap. The size of the heatmap depends on the number of values in the list, therefore allowing it to grow based on the user's input. The code in this file is inspired by NikKonst's 'ld_plot' tool (https://github.com/NikKonst/ld_plot).

The file 'LD_outputs.py' contains the three functions used in 'main.py' to calculate the LD values for all populations, create the heatmaps for each of those values, and output the values in a .txt file. These functions are called 'LD', 'heatmap', and 'writing_table_to_file'.

The first function in the file called 'LD' takes a list of rs IDs as input. It calls to the 'linkage_disequilibrium_calculation' function in 'LD_calculation_function.py' and works over every possible pair of rs IDs in the inputted list. Next, the second function, 'heatmap', creates heatmaps for the r2 and D' values for all three populations. It does so by calling to the 'LDheatmap' function in the 'LDheatmap.py' file. The result of 'heatmap' is 6 heatmaps every time the function is called, with the size depending on the input. An example of heatmaps for the British r2 and D' values is in Figure 7.

Finally, the final 'writing_table_to_file' function creates a tab separated text file. In 'main.py', this function is used to create the txt file for all the dictionaries outputted by 'LD'.

The user has the option save the .txt file of the LD calculations for the rs IDs it has chosen. The results of the calculations are ordered in the rows, starting from the first pair in the first row to the last pair in the last row. An example of the results for the British population is in Table 1.

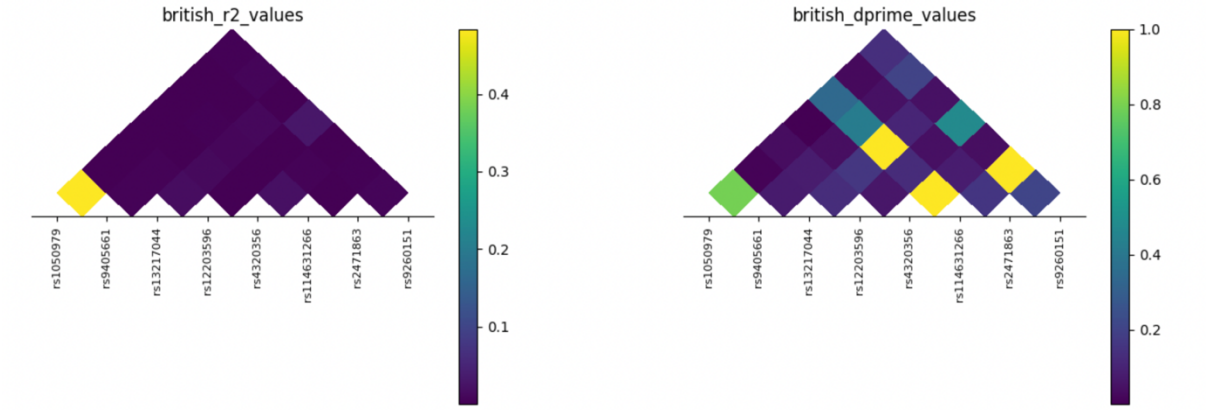


Figure 7. Example heatmaps for the British population r^2 and D' values for a query of 8 rs IDs. These are the first 8 rs IDs that are T1D relevant in chromosome 6. The bottom of the colour bar (dark blue) represents a low r2 or D' value. The top of the colour (yellow) represents a higher r2 or D' value, in comparison to the rest of the values in that dataset.

Table 1. Excel table view of the .txt file results from the query example in figure 7. Only the British population results are shown.

rs ID_1	rs ID_2	british_haplotypes	british_pA	british_pB	british_pAB	british_D	british_r2	british_Dprime
rs1050979	rs9405661	{'00': 92, '01': 9, '10': 22, '11': 77}	0.505	0.57	0.46	0.17215	0.48369786	0.792769975
rs1050979	rs13217044	{'00': 64, '01': 37, '10': 62, '11': 37}	0.505	0.63	0.32	0.00185	5.87E-05	0.00990099
rs1050979	rs12203596	{'00': 64, '01': 37, '10': 60, '11': 39}	0.505	0.62	0.32	0.0069	0.0008084	0.035956227
rs9405661	rs13217044	{'00': 75, '01': 39, '10': 51, '11': 35}	0.57	0.63	0.375	0.0159	0.00442495	0.075391181
rs9405661	rs12203596	{'00': 74, '01': 40, '10': 50, '11': 36}	0.57	0.62	0.37	0.0166	0.00477197	0.076638966
rs13217044	rs12203596	{'00': 84, '01': 42, '10': 40, '11': 34}	0.63	0.62	0.42	0.0294	0.015739	0.128160418

Limitations and future work

There are some limitations to our software. Firstly, the selection of rs IDs for linkage disequilibrium may not be the most user-friendly. With the current format, it will be quite tedious to copy/ type out many SNP names for linkage disequilibrium calculation. The use of check boxes along with a 'select all' button may make it easier for users to calculate many SNPs for linkage disequilibrium. However, this comes at a cost of limiting the user's SNPs input to only the SNPs present on the output page for linkage disequilibrium. A nice functionality to have would be for the user to view the LD measures for the whole chromosome.

Detection in the search bar can be further optimised. Searching a SNP only requires the first two character of what was inputted to be 'rs'. Potentially, inputs such as 'rsjkjnk' can be accepted and redirected to the SNP page despite not being a valid SNP name. As SNP names contain a set of integers after 'rs', an improvement on the detection of the search bar would be to detect the set of integers after 'rs'.

Additionally, our software is susceptible to SQL injection attacks. As what is typed in the search bar can be substituted into a SQL statement to search through the database, this makes it vulnerable for user to input their own SQL statement to add, delete, or access unauthorised data. However, with the criteria used in the search bar, this is less likely to occur. The implementation of SQLAlchemy may circumvent this issue.

On the Manhattan plot, some points are hard to discern from each other due to being close to each other on their location and p-values. To make it easier to differentiate between two similar points the Manhattan plot could be made more interactive. We can implement a solution where the rs ID label pops up when a user clicks on a specific SNP or even allow the user to click to show and hide SNPs. In addition, it would be useful to add a column with a link or reference to the study associated with those p-values.

The function called 'linkage_disequilibrium_calculation' in the LD_calculation_function.py file is quite big, with multiple functions inside it. This is not standard practice and not object-oriented programming, and it is difficult to test the functions within this function. We aim to rewrite this file in the future, so that this be reformatted to make the internal functions as standalone functions. We also aim to write unit tests for all code and add it to our repository.

The application's philosophy is for **all** Type 1 Diabetes SNPs to have relevant data. Currently our tool has limited data, primarily information on chromosome 6. This will be expanded to include the whole genome. However, by expanding to the whole genome, it is likely a different storage option would be utilised such as 'MySQL' as Sqlite finds it difficult to handle large datasets (Hipp, 2020). Our database also only includes CADD-PHRED scores to represent a measure of functional impact. We aim to add additional functional impact scores such as Polyphen and SIFT, as they are easily obtained with SNPnexus.

Lastly, our software only contains the genotype information for the GBR, ESN, and JPT populations. While we are using the data from three populations from different geographical locations, it is still not representative of all populations. Our next steps would be to include all 1000 genome populations, as an attempt of being more inclusive and representative.

Conclusion

In conclusion, we successfully developed a web browser that leverages upon genomic (SNPs, mapped genes), functional (gene functional terms), and SNP functional impact data (CADD-PHRED scores) available for type-1 diabetes. We also successfully created a tool that calculates linkage disequilibrium measurements for pairs of SNPs in the British GBR, Nigerian ESN, and Japanese JPT 1000-genome populations. The software also produces Manhattan plots for the chromosome regions searched by the user, and heatmaps for the linkage disequilibrium measures for the SNPs chosen by the user. We believe this tool has the potential to be used by researchers to interpret the mapping of the genetic basis of T1D.

References

Cunningham , James E. Allen, Jamie Allen, Jorge Alvarez-Jarreta, M. Ridwan Amode, Irina M. Armean, Olanrewaju Austine-Orimoloye, Andrey G. Azov, If Barnes, Ruth Bennett, Andrew Berry, Jyothish Bhai, Alexandra Bignell, Konstantinos Billis , Sanjay Boddu, Lucy Brooks, Mehrnaz Charkhchi, Carla Cummins , Luca Da Rin Fioretto, Claire Davidson, Kamalkumar Dodiya, Sarah Donaldson, Bilal El Houdaigui, Tamara El Naboulsi, Reham Fatima, Carlos Garcia Giron , Thiago Genez, Jose Gonzalez Martinez, Cristina Guijarro-Clarke, Arthur Gymer, Matthew Hardy, Zoe Hollis, Thibaut Hourlier , Toby Hunt, Thomas Juettemann , Vinay Kaikala, Mike Kay, Ilias Lavidas, Tuan Le, Diana Lemos, José Carlos Marugán, Shamika Mohanan, Aleena Mushtaq, Marc Naven, Denye N. Ogeh, Anne Parker, Andrew Parton, Malcolm Perry, Ivana Pili˘zota, Irina Prosovetskaia, Manoj Pandian Sakthivel, Ahamed Imran Abdul Salam, Bianca M. Schmitt, Helen Schuilenburg, Dan Sheppard, José G. Pérez-Silva, William Stark, Emily Steed, Kyösti Sutinen, Ranjit Sukumaran, Dulika Sumathipala, Marie-Marthe Suner, Michal Szpak, Anja Thormann, Francesca Floriana Tricomi, David Urbina-G˝omez, Andres Veidenberg, Thomas A. Walsh, Brandon Walts, Natalie Willhoft, Andrea Winterbottom, Elizabeth Wass, Marc Chakiachvili, Bethany Flint, Adam Frankish , Stefano Giorgetti, Leanne Haggerty, Sarah E. Hunt , Garth R. IIsley, Jane E. Loveland , Fergal J. Martin , Benjamin Moore, Jonathan M. Mudge, Matthieu Muffato, Emily Perry , Magali Ruffier , John Tate, David Thybert, Stephen J. Trevanion, Sarah Dyer, Peter W. Harrison , Kevin L. Howe , Andrew D. Yates , Daniel R. Zerbino and Paul Flicek. Ensembl 2022. Nucleic Acids Res. 2022, vol. 50(1):D988-D995 PubMed PMID: 34791404.

Grinberg, M., 2018. Flask web development: developing web applications with python, " O'Reilly Media, Inc."

Hill, W.G. and Robertson, A., 1968. Linkage disequilibrium in finite populations. *Theoretical and applied genetics*, *38*, pp.226-231.

Hipp, R.D., 2020. SQLite, Available at: https://www.sqlite.org/index.html.

Jorge Oscanoa, Lavanya Sivapalan, Emanuela Gadaleta, Abu Z Dayem Ullah, Nicholas R Lemoine, Claude Chelala, SNPnexus: a web server for functional annotation of human genome sequence variation (2020 update), Nucleic Acids Research, 2020, 48(W1):W185-W192.

Lewontin, R.C. and Kojima, K.I., 1960. The evolutionary dynamics of complex polymorphisms. *Evolution*, pp.458-472.

Marc Gillespie, Bijay Jassal, Ralf Stephan, Marija Milacic, Karen Rothfels, Andrea Senff-Ribeiro, Johannes Griss, Cristoffer Sevilla, Lisa Matthews, Chuqiao Gong, Chuan Deng, Thawfeek Varusai, Eliot Ragueneau, Yusra Haider, Bruce May, Veronica Shamovsky, Joel Weiser, Timothy Brunson, Nasim Sanati, Liam Beckman, Xiang Shao, Antonio Fabregat, Konstantinos Sidiropoulos, Julieth Murillo, Guilherme Viteri, Justin Cook, Solomon Shorser, Gary Bader, Emek Demir, Chris Sander, Robin Haw, Guanming Wu, Lincoln Stein, Henning Hermjakob, Peter D'Eustachio, The reactome pathway knowledgebase 2022, *Nucleic Acids Research*, 2021;, gkab1028, https://doi.org/10.1093/nar/gkab1028

NikKonst, ld-plot, 2021, Github Repository, https://github.com/NikKonst/ld_plot.

Safran M, Rosen N, Twik M, BarShir R, Iny Stein T, Dahary D, Fishilevich S, and Lancet D. The GeneCards Suite Chapter, Practical Guide to Life Science Databases (2022) pp 27-56

Rentzsch P, Schubach M, Shendure J, Kircher M. *CADD-Splice—improving genome-wide variant effect prediction using deep learning-derived splice scores.*Genome Med. 2021 Feb 22. doi: 10.1186/s13073-021-00835-9. PubMed PMID: 33618777.

Sollis E, Mosaku A, Abid A, Buniello A, Cerezo M, Gil L, Groza T, Güneş O, Hall P, Hayhurst J, Ibrahim A, Ji Y, John S, Lewis E, MacArthur JAL, McMahon A, Osumi-Sutherland D, Panoutsopoulou K, Pendlington Z, Ramachandran S, Stefancsik R, Stewart J, Whetzel P, Wilson R, Hindorff L, Cunningham F, Lambert SA, Inouye M, Parkinson H, Harris LW. The NHGRI-EBI GWAS Catalog: knowledgebase and deposition resource. Nucleic Acids Res. 2022 Nov 9:gkac1010. doi: 10.1093/nar/gkac1010. Epub ahead of print. PMID: 36350656.

The 1000 Genomes Project Consortium. A global reference for human genetic variation. Nature 526, 68–74 (2015). https://doi.org/10.1038/nature15393

Appendix

List of all the chr6 rs IDs in the database.

rs1050979
rs9405661
rs13217044
rs12203596
rs4320356
rs114631266
rs2471863
rs9260151
rs34941730
rs2523989
rs45626136
rs886424
rs118124843
rs116020851
rs1265057
rs116763857
rs9264758
rs2251396
rs9501130
rs3093664
rs2857595
rs1270942
rs3134938
rs926592
rs1980493
rs3135365
rs9268576
rs9268645
rs9268853
rs138748427
rs7760731
rs9272346
rs9273363
rs9273368
rs1770
rs9275358
rs2647044
rs3135002
rs35122968
rs3997848
rs1015166
rs241427
rs4148870
rs3128930
rs3761980
rs80028505
rs78824139
rs17711850

rs3757247
rs11755527
rs72928038
rs9354144
rs11154178
rs9388489
rs1578060
rs1538171
rs12665429
rs6931865
rs212408
rs9356171
rs73043122
rs924043