Documentation

About

With increasing understanding of genome research and the increasing statistics associated with diabetes, the need for a comprehensive diabetes genome platform is in need. “Type 1 Diabetes SNP Portal” is a fast, efficient, and user-friendly website allowing individuals to easily extract data associated with genetic variants related to type 1 diabetes. The “Type 1 Diabetes SNP Portal” was developed by 4 students at Queen Mary University of London, under the supervision of Matteo Fumagalli and Conrad Bessant, both lecturers in bioinformatics at the university.

The search features of the website are linked to a database that contains broad information associated with the variants seen in type 1 diabetes across all chromosomes. This includes SNP names (rs values), genomic positions, p-values from the association test, mapped genes, variant allele frequency in British, Han Chinese and Yoruba populations. There is also gene ontology information as well as Relative Allele Frequency (RAF), clinical impact and relevance information. Linkage disequilibrium (LD) values and also Manhattan plots are also included.

Being user friendly the website contains multiple pages that enable extraction of the specific values mentioned depending on either variant ID, chromosome location, or gene name. The website allows further comparison of a linkage disequilibrium for each variant search as well as a linkage disequilibrium heat map plot.

Table of Contents

Software Architecture

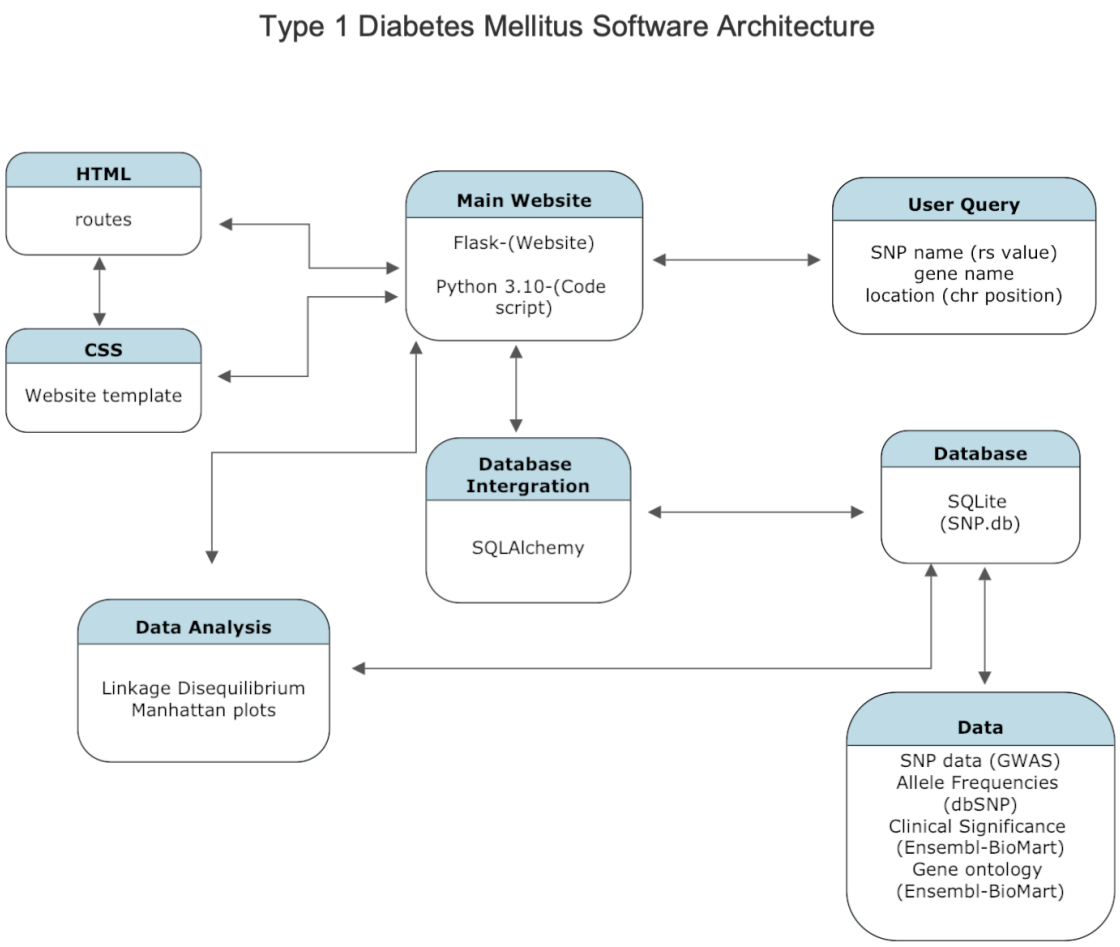

Figure 1: Shows Type 1 Diabetes SNP Portal software schematics and component integration.

The software architecture shows the main components and their integration with one another to produce the final functioning website. The website has two main components namely the Front-end and Back-end.

The Back-end components include the databases created using SQlite3 and all the integration software (SQLAlchemy). The database was populated using data collected from a variety of database websites including Genome Wide Association Study (GWAS), Ensembl-BioMart,

and NIH (National institute of healthcare). Due to a variety of data types, each data table was connected to Flask using SQLAlchemy, this is a python toolkit designed for database accessing that considers a familiar relationship within varying databases, as opposed to a collection of tables, simply put; all tables have a consistent relationship, which in this case is the variant ID or rs values.

Finally the Front-End component of the website contains routes defined using the HTML language and design layout using CSS that will be rendered by Flask templates. The user will interact with the Front-end to input their query and the website will send the query to the database and extract the relevant data by communicating with the Back-end components and send feedback to the user on the website Front-end. Data analysis will be carried out in the Back-end and created plots will be shown on the website pages.

Data Collection

SNP and genes data

Initial association data on type 1 diabetes mellitus was acquired from the GWAS Catalog (https://www.ebi.ac.uk/gwas/). The information contained in the catalog includes; single nucleotide polymorphism (SNP) variant (rs values), p-values from the association tests, mapped genes and chromosome location (position) for each variant. A comma separated values (csv) file was downloaded and this was used to create the main dataset for type 1 diabetes SNP. Additional information not included in the file (ie gene ontology , clinical relevance) were obtained from other databases such as the dbSNP form NCBI, Ensembl and IGSR.

Clinical significance

Clinical relevance data for each chromosome was obtained from the Ensembl database (https://www.ensembl.org/index.html) using BioMart. The Ensembl Variation 108 database and the human short variant dataset that includes SNPS and indels (insertions and deletions) (GRCh38.p13) was selected. For each query, data was filtered using chromosome number and a list of variants (rs values of interest). Information collected included variant name, chromosome location, clinical significance, variant and allele consequence, polymorphism phenotyping version 2 (PolyPhen-2) prediction and score, sorting intolerant from tolerant (SIFT)

prediction and score. SIFT (https://sift.bii.a-star.edu.sg/www/SIFT_dbSNP.html) and Polyphen-2 (http://genetics.bwh.harvard.edu/pph2/) are tools that predict whether amino acid substitution affects protein function and the impact of that substitution respectively. In clinical significance,variants are described as benign (not disease causing) or pathogenic(disease causing). Polyphen describes SNPs predictions and scores by severity of damage ranging from 0 (benign/tolerated) to 1 (damaging/deleterious). SIFT predicts whether a SNP is tolerated and has no effect on function or deleterious (affects protein function).SIFT score ranges from 0 (deleterious) to 1 (benign) with a threshold of 0.05 (score <0.05 is considered deleterious).

Allele Frequencies

Obtaining genome wide data from GWAS aided in then searching for the necessary Relative Allele Frequency (RAF) data. RAF data was collected using a combination of the Genome Wide Association Study (GWAS) database (Genome-Wide Association Studies (GWAS)), and the Ensembl database using BioMart (Ensembl genome browser 108). Obtaining the RAF data was done manually. BioMart provides community-based data which includes RAF for certain populations of continents including European sub populations, Asian sub populations, and finally African Sub populations, respectively those of Great Britain, Han Chinese, and Yoruba. The GWAS data obtained indicated alleles associated with the disease (A, T, C, G); this aided in two things, the first being an indication if the relative allele frequency was even known for that variant, and if it was what the relative frequency, because of the the RAF data collected for all chromosomes is slightly less than that of the variant IDs provided.

Gene Ontology

Gene Ontology terms were obtained from Ensembl using BioMart (https://www.ensembl.org/index.html). The Ensembl Genes 108 database and human genes (GRCh38.p13) dataset were selected. A list of all the associated mapped genes were inputted and the output was filtered to show the gene name along with the corresponding Gene Ontology (GO) domain, GO term name, GO term definition and GO term accession numbers. However, the ontology for some of the genes were missing. For some of these genes the ontology was obtained through the Gene Ontology database (http://geneontology.org) using the AmiGO 2 tool (http://amigo.geneontology.org/amigo/landing) and excluded GO terms with missing ontology data.

Database Schema

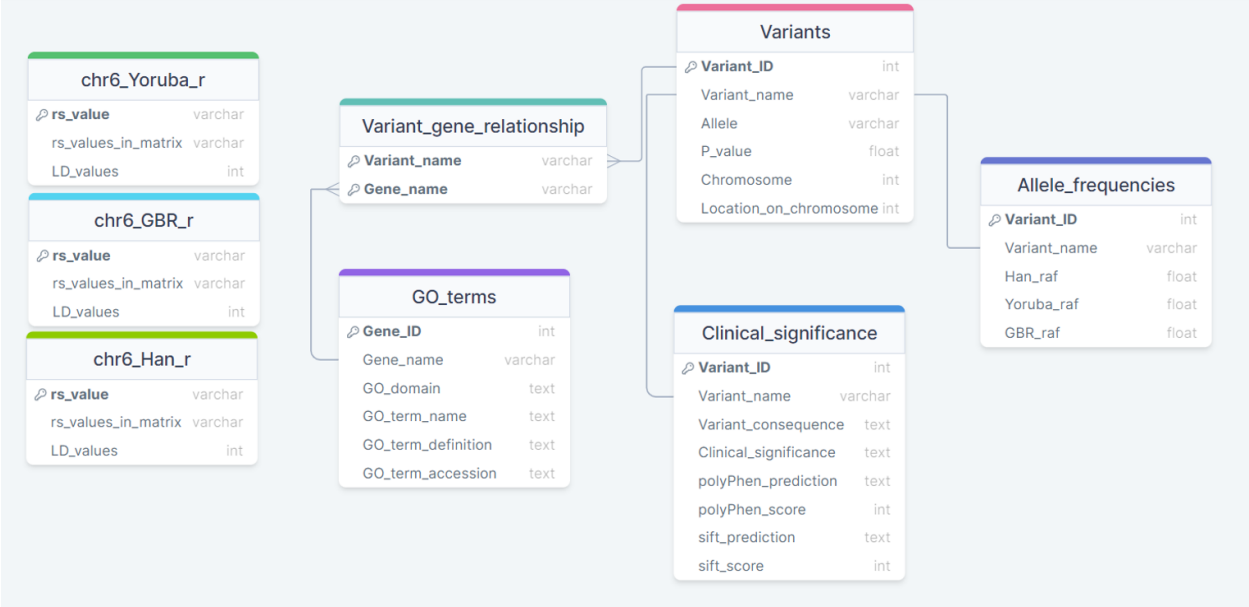

Figure 2: Shows the relational database schema for the website

The database schema above represents a SQLite3 database which we produced to limit the repetition of rows due to one-to-many relationships between data entries, this is a widely acknowledged benefit to the use of relational databases. The main table is the variants table which takes data from GWAS Catalog and holds basic identifying information on the SNPs associated with T1 Diabetes. Three tables were defined to hold information used for the results of queries for either variant name (rs value), gene name, or chromosome location; these are Allele_frequencies, Clinical_significance and GO_terms. Allele_frequencies and Clinical_significance have variant_name foreign keys that relate to the Variants table, GO_terms required a Variant_gene_relationship table that has foreign keys making a one-to-many relationships with GO_terms and Variants which deals with the issue of the many-to-many relationship between variants and genes in our acquired data i.e. rs values are associated with multiple genes and genes all have multiple associated rs values. When a search using a variant name is performed on the website the Variants table is joined to the other tables to return information to the website for allele frequencies, clinical significance and go terms. When a search is performed using a gene name, the variant_gene_relationship table is joined to the

clinical significance and allele frequency tables to return data for these categories filtered by gene name, go_terms doesn't require a join for this. Searches performed using chromosome location require joins of the Variant table with the other tables to allow filtering to be done by location, similar to the method performed with searches using variant name. The three chr6_*_r tables are matrices containing Linkage Disequilibrium data that did not require any relationships between tables to be formed.

Website Architecture

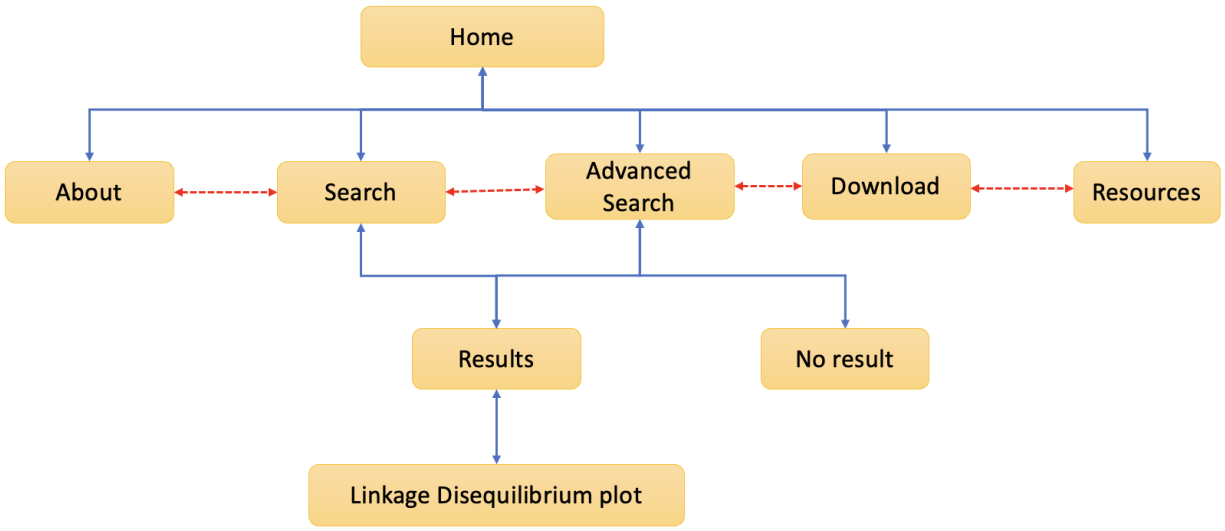

Figure 3: Overview of the website architecture. The blue lines link to web content and the red dashed lines are internal links.

The layout structure of the website and how each page is connected is shown in figure 3. The homepage has direct links to five routes namely; the About, Search, Advanced Search, Download and Resources hyper-links. Internal links shown as red dashed links allow the routes to be directly connected from one route to another without accessing the homepage route. From the search routes, the website will direct to the results route if the query is in the database. Alternatively, if the search is not in the database, the website will redirect to a no result page. From the results route, additional analysis produces a plot that links to the LD plot route with the displayed results.

Website Layout and Design

The structure that defines every component of each webpage was designed using hyper-text markup language (HTML) tags. HTML pages allow Flask to specify the routes for each page. The cascading style sheets (CSS) were used alongside HTML to provide the website with appealing visual appearance with a professional look and interactive elements. These css sheets were saved separately and linked to the respective HTML page in the header section. Jinja, a web template engine for python, that was used as 1) creating a loop function for adding table rows as well as 2) extending the base HTML page design to all pages.

Website Features

Homepage design

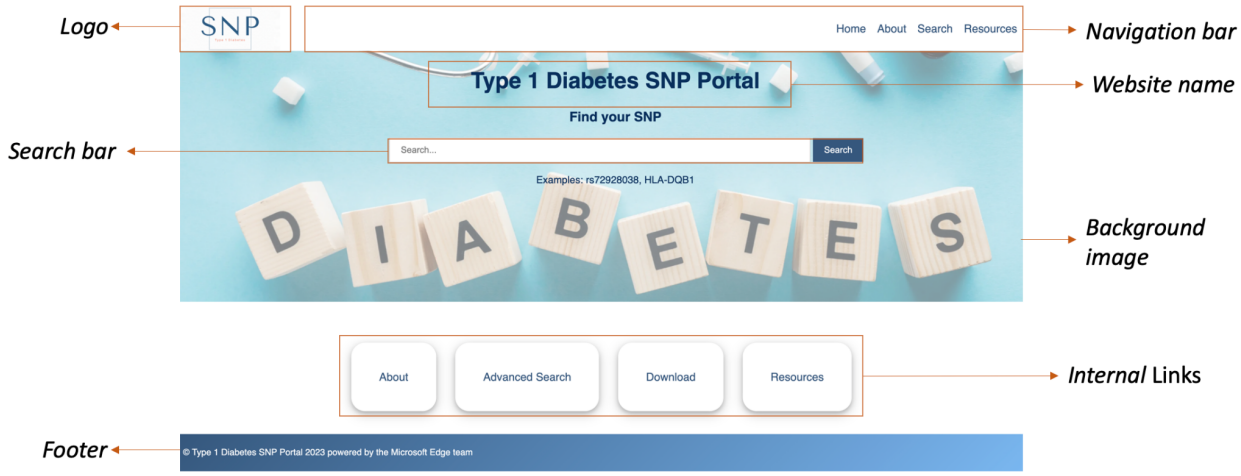

Figure 4: Shows the homepage features for the website front-end

The website contains a top section that contains the SNP website logo and navigation bar. The navigation bar has internal links to the Home, About, Search and Resources pages and this feature is contained in all the webpages. The next section with the diabetes background image contains the website name and the search bar. Users input SNP and gene queries in the search bar and by submitting the query, it connects it to the database to search relevant information using SQLAlchemy. The next section of the webpage contains internal links with similar

functions as the navigation bar, with an additional download link. The final section is the footer section that is also contained in all the webpages.

About Page

In addition to the navigation bar and the footer, the About page has the summary about the website and some information about type 1 diabetes. Next, the page has contact information for which users can contact the authors of the website. Furthermore, this page contains some relevant citations and acknowledgements.

Internal Web Pages

Advanced Search: This option allows the user to specify the search query as either SNP, gene or genomic coordinates. Users cannot search with genomic coordinates on the search bar in the homepage as this requires two inputs (range from position *x* to *y*). If users search for genomic coordinates in the search bar, the website will redirect to a no result page and provide a suggestion of the required search format. A link to the advanced genomic page is also provided to make the website user-friendly.

Results Page: Output from a search query is displayed in the results page. The page will return the user's query as the header and several tables containing variant information, clinical significance, and gene ontology terms and description. In addition to the tables this webpage also contains a Manhattan plot for type 1 diabetes SNPs from GWAS studies. Users are able to interact with the plot, and the query SNPs will be highlighted. Users interested in Linkage disequilibrium analysis need to fill in a form in the website that specifies the population of interest and a list of SNPs found in the search query. Once these conditions have been met, the user can submit and a heat plot showing the distances will be created. The default population is the British, therefore if the user does not specify the population, data will be extracted from the British RAFs. Alternative population options include Han Chinese and Yoruba. The results page requires scrolling since the information display takes up more than one page. To facilitate ease of navigation, there is a side navigation bar which allows users to jump to certain parts of the page for quick access.

Download: The download option will redirect a user to the download page that contains the data used to create the databases. The users can download the dataset of interest for further research and a brief description of the dataset is provided in the download table as well as the data formats.

Resources: For additional information about the data itself, there is a resources option that users can access from the homepage and/or navigation bar. The page contains useful resources that were used for data collection and have additional function and data outside the scope of this website. This allows users to obtain all the information and tools they might need by clicking the Learn More button which contains external hyperlinks to respective databases and tools.

Back-end Administration

All the functionality and communication between the backend and frontend is facilitated by Flask. Information from a query is sent to the database and queried using SQLAlchemy and feedback is sent to flask which displays the information by rendering the required templates and routes from HTML/CSS. This allows for a seamless experience for the end user. Data analysis that creates the (Manhattan and LD) plots is carried out in python and results are rendered to the website in the same manner. Separate python files were created to compartmentalise the code and ensure it is not all ran at once: main.py contains all of the individual Flask routes, models.py contains SQLAlchemy ORM models that were used to map the database, functions.py contains almost all functionality including the search & query functionality, app.py initialises the flask app and connects it to a database engine, init_db.py initialises the db and creates the SQLAlchemy engine and session.

Data Visualisation

Manhattan plot

The R package “manhattanly” was used to create interactive manhattan plots. The x axis represents the chromosomes and the y axis the -log10 of the p-values. Each point represents a

SNP and all the highlighted SNPs represent the SNPs returned from the query. By hovering over points on the plot, users can see details that include the SNP name and the mapped gene. Dragging a rectangle around an area of the plot will zoom into the area to have a better view of the points on the manhattan plot.

Creating a Manhattan plot using only SNPs returned from a genomic coordinate search was not ideal as only a small number of SNPs would be used to make the plot, therefore an interactive Manhattan plot with all the SNPs was more reasonable. This is why “manhattanly” in R was used since this package creates an interactive Manhattan plot and allows you to highlight specific SNPs. This allows the user to view all the SNPs associated with Type 1 Diabetes in a Manhattan plot whilst being able to identify the specific SNPs that were returned in the query. Since the plot is interactive it also makes it easier for the user to view individual SNP names and their mapped genes without having labels overlapping each other making it difficult to read. In order to integrate the interactive Manhattan plot into the results page, an R function was written where the argument of the function is the list of SNPs that need to be highlighted and after calling the function a html of the interactive Manhattan plot is created and saved to a file. The python package “rpy2” was used to call the R function in a python environment and ultimately create the interactive plot. The interactive plot html was then linked to the results template which allows users to view the interactive Manhattan plot after they search a query.

Linkage Disequilibrium Plot

The python package called ld_plot is available for any versions of python > 3.7, this was used for plotting the linkage disequilibrium (LD) plot. The function of the plot is to take the LD values between the SNPs selected on the webpage, in this instance these values are already calculated, and were obtained from the National Cancer Institute (NIH) website on the LD link page, if only one SNP is selected, the graph produced will be non existent. Given that the r^2 values were already calculated, a matrix was produced using python for the values between SNPs that can be called within the ld_plot function.

The website allows a selection of SNPs from only chromosome 6 (as this is the chromosome associated with the greatest amount of data in relation to Type 1 Diabetes), once these have been made the plot will be produced. The plot is a triangular heat map that has been rotated so that the only axis is at the bottom. The colour of different cells within the heatmap denotes the

value of r^2 as indicated in the colour key; in this instance the closer to 1 the r^2 value is to 1 the colour of the square will appear yellow, and the closer the value is to 0, the colour will appear blue.

Deploying Type 1 Diabetes SNP Portal to the web

Amazon Web Services (AWS) has several cloud services that can be utilised to deploy web applications. The Amazon Elastic Compute Cloud (EC2) service was used to deploy our web application. EC2 is a virtual server that allows you to run applications on the AWS cloud. An Instance (a virtual machine) was created using EC2 and in this case a windows template was used due to familiarity however other templates such as linux and OS can be used. While creating the instance, the network settings had to be specified. http and https traffic from the internet was allowed so all IP addresses can access the instance. This allows anyone to access the website.

Deploying the web app via a virtual machine was ideal since the web application requires both python and R to run since the Manhattan plot in our website requires R to function and the rest in python. This is why EC2 was used instead of a service such as Elastic Beanstalk, which is made specifically for website deployment. Elastic Beanstalk requires the language your application uses to be specified and therefore is unable to incorporate R into this. After connecting to the EC2 instance both python (3.11.2) and R (4.2.2) were downloaded along with the python packages required for the application to run which is found in the requirements.txt file. In the main.py file that is used to run the application, the host “0.0.0.0” had to be specified and port “80” (the default network port that is used to send web pages and also receive them) so it can be publicly available. The flask application was then run, ultimately making the website available on the server. The public IP address for the instance that was created using EC2 on AWS now contains the website and by copying http://13.42.53.58 into a browser, anyone is able to access it.

Running the website locally

To locally deploy the website, make sure that all imports are downloaded onto your machine with the correct versions (refer to requirements.txt). Then in the terminal navigate to the final_website folder and run “python main.py”. Then enter the provided URL into a browser of your choice.

Limitations

Data scarcity, particularly clinical and functional data posed as a limitation to the website. This is because finding ways post-GWAS to translate genetic associations into clinically useful information is challenging.

Limitations in being unable to download Variant Call Format (VCF) file from the International Genome Resource Sample (IGSR) meaning that the LD values were obtained manually, this requires extensive amounts of time and manual adjustments, because of time restrictions, in creating a ld database based on SNPs and then producing a matrix that suits the ld_plot function, we were only able to provide ld values for chromosome 6 and thus only produce the plot for chromosome 6.

Future Developments

Using application programming interface (API) to create the website that has the latest and up to date information.

Expand the linkage disequiibrium data to include the entire genome and expand to other populations

Expanding the screen resolution to work for all screen types

References

Nyaga DM, Vickers MH, Jefferies C, Perry JK, O'Sullivan JM. The genetic architecture of type 1 diabetes mellitus. Mol Cell Endocrinol. 2018 Dec 5;477:70-80. doi: 10.1016/j.mce.2018.06.002. Epub 2018 Jun 18. PMID: 29913182.

Slatkin M. Linkage disequilibrium--understanding the evolutionary past and mapping the medical future. Nat Rev Genet. 2008 Jun;9(6):477-85. doi: 10.1038/nrg2361. PMID: 18427557; PMCID: PMC5124487.

Freedman ML, Monteiro AN, Gayther SA, Coetzee GA, Risch A, Plass C, Casey G, De Biasi M, Carlson C, Duggan D, James M, Liu P, Tichelaar JW, Vikis HG, You M, Mills IG. Principles for the post-GWAS functional characterization of cancer risk loci. Nat Genet. 2011 Jun;43(6):513-8. doi: 10.1038/ng.840. PMID: 21614091; PMCID: PMC3325768.

On beyond GWAS. Nat Genet. 2010 Jul;42(7):551. doi: 10.1038/ng0710-551. PMID: 20581872.