ⓒCopyright 2020

Yiliang Ma

# Everyone’s Variant Annotation Tool: EVAT

Yiliang Ma

A thesis

submitted in partial fulfillment of the

requirements for the degree of

Master of Science

University of Washington

2020

Committee:

Sean Mooney

John Gennari

Program Authorized to Offer Degree:

Department of Biomedical Informatics and Medical Education

University of Washington

## Abstract

Everyone’s Variant Annotation Tools: EVAT

Yiliang Ma

Chair of the Supervisory Committee:

Professor Sean Mooney

Department of Biomedical Informatics and Medical Education

Currently, there is a lot of genetic variant information distributed in many different databases, and it will cost individuals plenty of time to retrieve data from those resources.

In this thesis, I develop EVAT(Everyone’s variant annotation tool), a tool aiming at helping individuals retrieve annotation information about their genetic variants. People with or without programming skills may choose different methods to get their genetic variant annotation information. For individuals who have program skills, EVAT offers Python APIs which connect to the backend directly to help them retrieve annotation information. The backend of the tool is built by four file interpreters that translate file format, a module that sends and receives information from MyVariant.info, a module that converts the JSON result from MyVariatn.info to Panda dataframe and three functions that support different queries. For individuals who don’t

have program skills, EVAT offers a graphical user interface, which is the front end of the tool. This graphical user interface allows users to upload files, read the annotation, and do the query by mouse so that they don’t need coding skills when doing genetic annotation.

EVAT can be used either as a backend only( for users with programming skills) or with a graphical user interface to easily query and retrieve annotation information. This annotation information could help people understand the effect of genetic variants  or do further research about them.

# Table of Contents

## ACKNOWLEDGMENTS

First and foremost, I wish to express my sincere gratitude to the University of Washington for providing access to amazing faculty and world-class research facilities/resources. During these 3 years, I learned not only the knowledge but also ways of doing research. This thesis could not be done without the support of many people.

My mentor Sean Mooney, thank you for being strict and your group affects me from every perspective. Your way of doing research lets me experience what attitude and what effort I should have when doing real research.

My committee member John Gennari, I really appreciate your suggestions on my thesis and project. You are very patient and kind to guide me how to do research and write a thesis.

Postdoc in my group: Vikas, thank you for spending a lot of time guiding my thesis, including the plan of the whole idea, how to do the research and the writing skills. Furthermore, you give me suggestions on my career and exchange views about events in life, which provide a positive effect on my whole life.

Finally, I would like to personally acknowledge some incredible people: my peer Jimmy who helped me a lot about writing and presentations, My friends Chenchao Xu, Sicheng Song and Yiqun Rong give me a lot of advice on courses and life in BIME, Lora and Jill who answer my questions about study in BIME, my friend Yuanze Zhang, all in Sean Mooney’s group and my parents.

Chapter 1

# INTRODUCTORY STATEMENT

## *1.1 Goal of the Thesis*

EVAT(Everyone's variant annotation tool) is a tool that aims at helping everyone get their genetic variants annotation. Individuals may have different backgrounds, some of them may have program skills and some of them not. General public may have their own variant sequences, like the genetic sequence file 23andMe. However, they need tools to let them know what the variant is about, like the variant is about the color of your eyes or the color of your hair or something else. ("The Genetics of Blond Hair" 2014) Biologists may have sequences about many human tissues and may want to know what we have already known about these variants and where they could do more research. A bioinformatician may want to get the variants information of large files to do their own research. EVAT could help everyone get their variant annotation

With four kinds of file input recognition, three query functions and two files output format and one graphical user interface. The EVAT system queries information from MyVariant.info databases which contains information from twenty databases, including large integrated databases and small but specialized databases. (Xin et al. 2016)

The work of this thesis is development of a tool that could help people with or without programming skill get their variant annotation information. The backend of the tool is built by four file interpreters, a module that could send and receive information from  MyVariant.info, a module that could convert the JSON result from MyVariatn.info to Panda dataframe and three functions to helper users do queries. The frontend of the tool is built by a group of modules that put close widgets into groups, build collapsible boxes, assign the location of the group and the location of widgets in the box and several modules that assign the relationship between the frontend button and backend.

Overview of the thesis: Chapter1 will introduce the tools, concept of variant and variant annotation; Chapter2 will introduce the problem we face, the related tools and MyVariant.info; Chapter3 and Chapter 4 will introduce the backend and the frontend of the tool; Chapter 5 is the discussion of the advantages, disadvantages, future plan of the tool and other expectation of the tool; Chapter 6 is the brief summary of the whole thesis.

## ***1.2 Variant and Types of Variants***

The variant is an alteration in the DNA nucleotide sequence. (“Definition of Variant - NCI Dictionary of Cancer Terms” 2012) Based on the different modifications to the DNA sequence, we have SNV, Indel, Copy Number variations and  Translocations and Inversions.

A single nucleotide variant (SNV) is the most common type of variant. SNV is the substitution of a nucleotide that occurs at a specific position in the genome, that leads to the change of gene sequence, influencing transcription and then translation. (Seeb et al. 2011) Therefore, different gene products (e.g., proteins) could be observed as a consequence of SNVs, which can cause different phenotypes or molecular disruption.  Sometimes these variants can have negative effects such as complex disorders or severe diseases. For example, red-green colorblindness is caused by mutations on the X chromosome such as Retinitis pigmentosa-10 (RP10). (Bowne et al. 2002) Until now, researchers have done a lot of work about variants, including variants' physical features, for instance, the location of variants and the frequency of the variants, the function to the biology process in human being, like how could this variant affects proteins and biology process that proteins take part in.

Also, not every SNV will cause the change of protein structure.(Figure 1.2.1) Many nucleotide mutations will not change the amino acid sequence and those are called Silent type of SNV. For instance, TTC and TTT are both code for Lys. Therefore, the protein structure doesn't change. Nonsense means a codon is changed into a stop codon, which will result in a truncated protein. This may change the protein function but probably not at the same time. For example, TTC changes to ATC and ATC is a stop codon. Also, there is one type of SNV called Missense, which means the sequence change does change the protein structure. For example, TTC changed to TGC. After the change of protein structure, the function of the protein may be changed. If the mutation happens in the important location of the protein sequence, the protein may lose all function, which will bring a negative effect on the normal biology process and might cause disease.

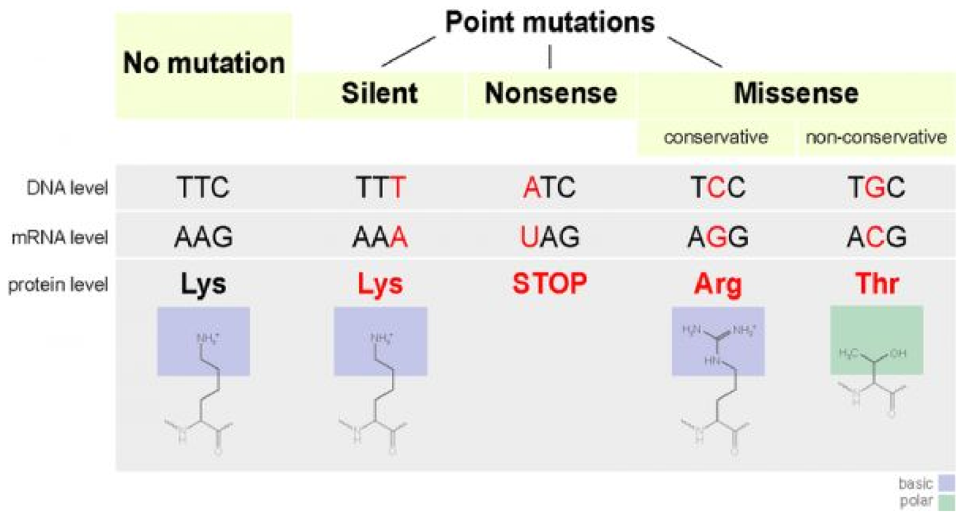


Figure 1.2.1: Mutations are classified based on how they affect the codon they are in. Image source: Wikimedia commons. (“What Effect Do Variants in Coding Regions Have?” 2019)

Indels is the term for genetic insertion and the genetic deletion. Genetic insertion is one type of variant, which means one or more bases are inserted into the original sequence. Deletion, on the other hand, means the one or more bases are deleted compared to the original one. It is estimated that there are millions of indels mutation and some of them may play roles in diseases.(Lin et al. 2017) For instance, indels are the leading cause in Cystic Fibrosis. (Chung et al. 2012) Cystic fibrosis is caused by CFTR gene three-base-pair deletion. (Trujillano et al. 2013)

Translocations and inversions are another two types of  variants. Refers to the gene rearrangements or the DNA segments. The DNA segments are broken off and if  some of them are located at other places of the chromosome, this is called translocation. if the DNAsegments are broken and some segments are reinserted into the chromosome with 180 degree reverse, that is called inversinos. Generally, the larger the DNA sequence is, the more likely Translocations and inversions happen.

Copy number variants are the many copies of a certain trait found in the genome. According to the central dogma, there are two copies of the gene in the human genome. However, there might be situations that many copies of a gene are shown in one's genome which causes health problems. Also, if there is only one copy or even no copy of the certain gene, that might likewise cause problems.

## *1.3 Variants Annotation*

Variant annotation is the process of assigning additional biologically meaningful information to a variant that goes beyond simple positional information, including function, frequency, disease association, etc. For example, assume someone gets a variant rs28931595 from a genetic test file. Without annotation, what he could know is the id of this variant only. However, after the variant annotation, after retrieving variant annotation information, they can find additional information such as the frequency of this variant, or the likelihood that the variant is associated with a specific disease or disorder.

One important type of annotation is assigning a functional consequence to a variant. Functional consequences caused by the variants are often defined by the physiological phenotypes. The genetic variants will cause many differences between humans. Some of them will not cause disease directly but some of them will cause disease or even death. There are five levels of clinical functional consequences represented by a simple controlled vocabulary. They are benign, likely benign, uncertain significance, likely pathogenic, pathogenic, where benign is that this variant will not cause disease and pathogenic is that this variant will cause disease.

Based on different features, there are three kinds of annotation query functions: the gene-based annotation, region-based annotation and filter-based annotation. Gene-based queries select only variants that belong to certain genes.. A variant is located on the chromosome and the location might belong to one or more genes. Region-based annotation is a method to help users identify whether the variant is located in a certain region of the chromosome. Filter-based queries accept a range and a field, and then the program will find variants that have the field in that range. For instance, the user could filt the variants that have frequency between 0.2 and 0.8 by the filter-based query.

Variant annotation could help researchers save their time. For example, the biologist gets six samples of cancer tissue and adjacent tissue from the hospital and they want to figure out which gene is related to cancer. Then they use next generation sequence technology and get a lot of variants. And based on the different expression level(different mRNA numbers between health tissue and cancer tissue), they got 50 variants that might be important to the cancer pathway. However, they know nothing about the variant. Therefore, they get the variant annotation and get that 38 variants have been fully researched and 12 of them are worthy to do further experiments. Also, they know more about these 12 variants like the chemistry features. Then they could design next experiments with the annotation information, like the relationship of these 12 variants and the pathway of cancer.

**Chapter 2:**

# RELATED WORK

### *2.1 Variant Information Availability*

Currently, there is a lot of variant information distributed in many different databases, and individuals can not access that data easily.(Arnold et al. 2015) Because every database could not cover all information of a variant, if someone wants to get several aspects of the variant, they need to go to many databases to search for the information they need. For example, if we need to know the information of variant ‘chr6:g.152708291G>A’, including the general frequency, the relationship between this variant and cancer, and proteins related to this protein. We need to go to cadd to get the frequency of the variant, CGI to get the cancer related information and Uniprot to get the protein related information. There are many different data resources such as dbSNP and ClinVar, which have different aims and focus on different aspects of genetics. (Sherry et al. 2001; Landrum et al. 2016). Some databases are large and integrated and some of them are small but focus on specific areas.

For example,  dbSNP is a large database that has a catalog of genome variation to address large amounts of sampling to help studies, gene mapping and evolutionary biology. dbSNP divides varians in 4 types and single nucleotide variants take 99.77%. Dbsnp is to facilitate large-scale research in genetic fields, evolutionary biology and physical mapping.

Also, there are some small data sources that focus on certain areas. CGI(Cancer Genome Interpreter), for instance, is a data source maintained by several clinical and scientific precision oncology researchers. This data source focuses on the relationship between variant and tumor type and the interaction with drugs, which is to help identify which tumor alterations may shape the response to anti-cancer therapies. The CGI applies two data sources to explore the relationship between gene alterations and drug response. One is Cancer biomarker, The other is cancer bioactivities.

However, there is no possible way that we could gather all information from a single database. Three kinds of fields might be easy to get, but how about three fields in 20 different databases? Also, researchers have to suffer from the various file formats and different reference genome versions of the data. Due to that situation, we would like to provide tools to individuals to gather all this information at the same time in order to see the whole picture, such as knowing the physical features and various biology fields at the same time. Thus, providing a tool to let people understand and interpret variants is important.

## *2.2 Annotation Tools*

ANNOVAR: ANNOVAR is a tool to functionally annotate genetic variants detected from diverse genomes with various functions. However, Annovar requires the user to download all files to local as Annovar uses all data based on local file so it takes plenty of space. Also, Annovar is unfriendly to users without a programming background. Users are required to use terminal knowledge to understand and take advantage of this tool. MyVariant.info solves this problem. (Li 2011)

SnpEff: SnpEff is a software that could annotate the variants via genomic location and predicts the coding effect. However, the SnpEff is based on sequencing information rather than variants location. For example, SnpEff's input is like "ACGTTGACCCTA" rather than location and chromosome. Also, the SnpEff will not gather information from other sources but predict by itself. All in all, SnpEff is very different from MyVariant.info by the information sources. (Cingolani et al. 2012)

OpenCravat: OpenCravat is a highly customizable decision support framework that could support variant and gene prioritization. It could help people with variant annotation but still, it requires data download to annotate the data and background to select suitable annotation files in their store, which is not friendly to the general public. (Douville et al. 2013)

The Ensembl Variant Effect Predictor is a toolset that could do analysis, annotation, and prioritization of genomic variants in coding and non-coding regions. However, it requires professional background and file download to annotate corresponding data. (McLaren et al. 2016)

However, the tools mentioned above require an expertise to use and require the downloading of many files and some of them are not user-friendly or easy to understand for people without a programming background. Therefore, there is a need to serve these different groups of users and help them retrieve and interpret variant data from databases.

## *2.3 MyVariant.info*

But there is one tool that doesn't have those disadvantages mentioned above. It is MyVariant.info. MyVariant.info is a web service for querying human variant annotation by collecting information across many resources. The variant annotations are structured research findings (e.g., chromosome and position variant located at, corresponding genes, related pathways) and they distribute in different databases. MyVariant.info could integrate them together. (Xin et al. 2015)

MyVariant.info integrates data from different data sources, then interprets and stores them into its own database. MyVariant.info integrates data from about 25 sources including large databases such as dbsnp and some small data sources such as Wellderly. For each data source, MyVariant.info has a unique importer to convert the data from resources into JSON file format. Nomenclature from the Human Genome Variation Society (HGVS) is used as the primary key for each variant. The output of the parser is stored into a MongoDB and the objects with the same primary key are integrated as an object that contains all fields of this variant.

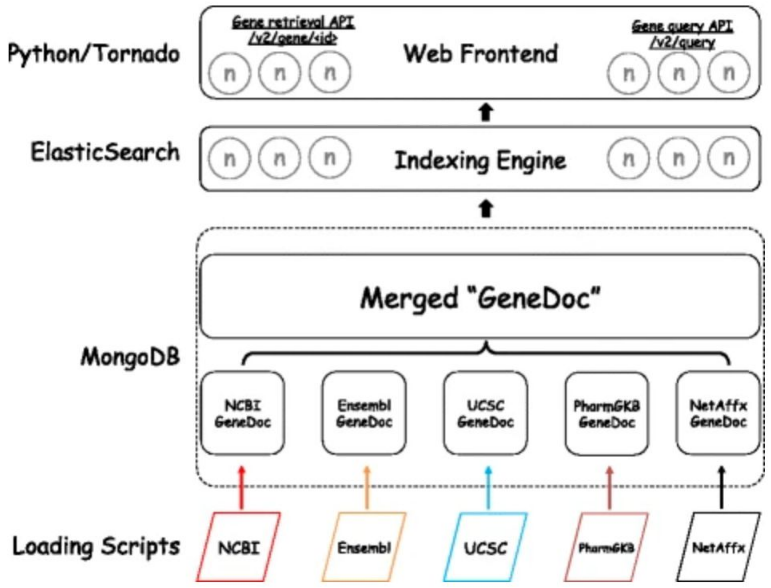
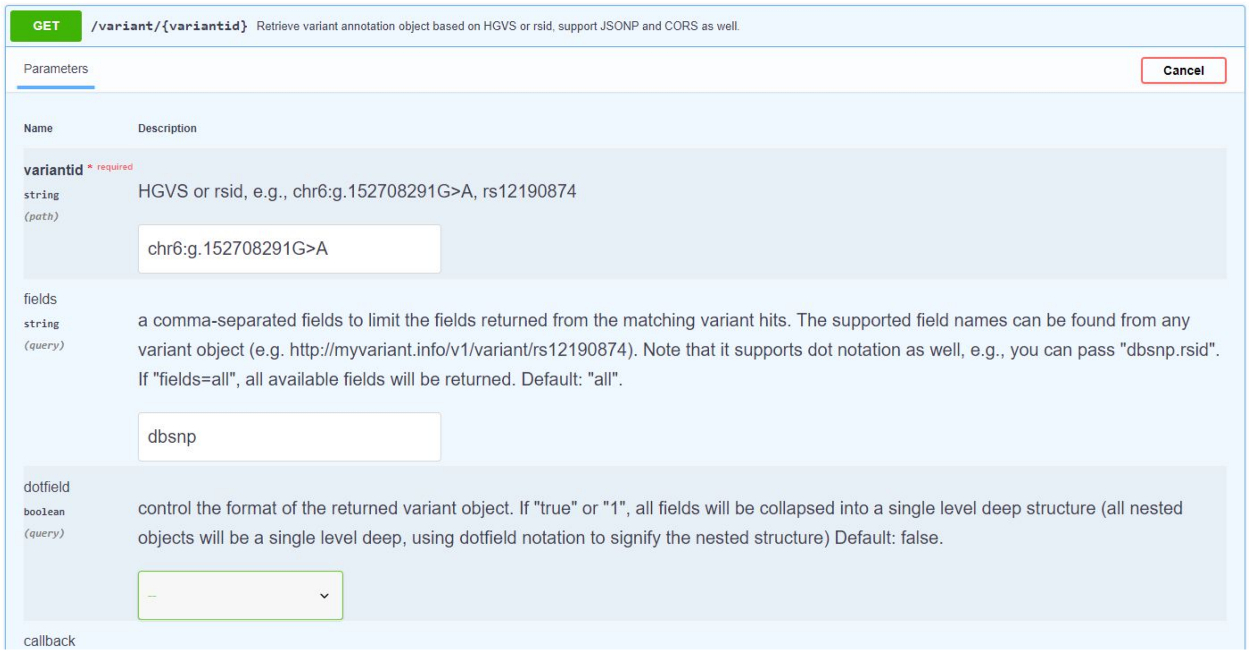


Figure2.3.1 The pipeline of the MyVariant.info.

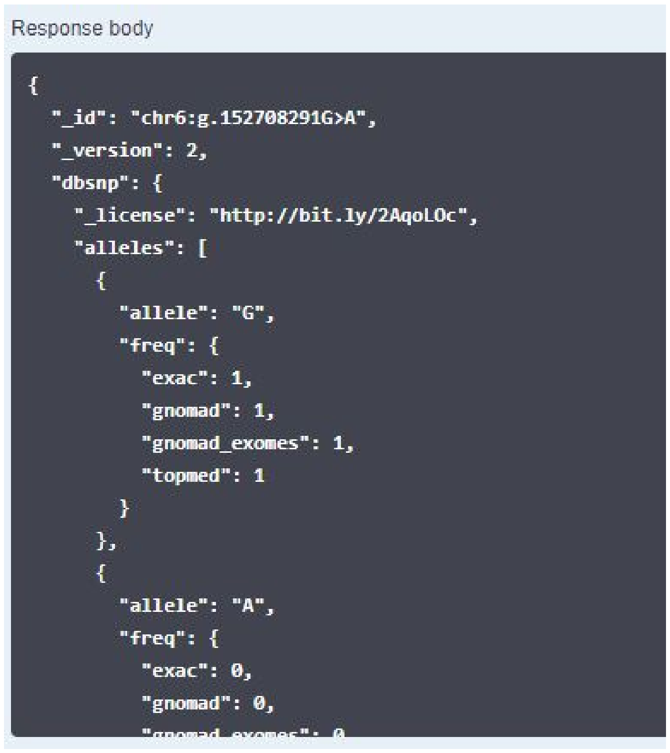
The web services could help users query information they need by various functions and specific parameters. After getting the input information, the indexing engine provides queries based on the primary key offered by users in MongoDB. Then, the output result shows up on the website for users.

To demonstrate MyVariant.info, a simple demo is listed in figure 2.3.2. Figure 2.3.2 (a) shows the query input website. Here we can select functions, (getVariant is selected here(top left)) and input the parameters we want. We use the chr6:g.152708291G>A is used here. Chr6 is chromosome number, 152708291 is position, G>A is genotype. And assuming we just want dbsnp(a famous database) information about this variant, we put “dbsnp” on the field textbox and then execute query.Then we get a JSON file(Figure 2.3.2(b) shows it) and a console respond that tells us the overview of the query.



`/variant/{variantid}` Retrieve variant annotation object based on HGVS or rsid, support JSONP and CORS as well.

| Name | Description |
|---|---|
| **variantid** * required<br>string<br>*(path)* | HGVS or rsid, e.g., chr6:g.152708291G>A, rs12190874<br>chr6:g.152708291G>A |
| fields<br>string<br>*(query)* | a comma-separated fields to limit the fields returned from the matching variant hits. The supported field names can be found from any variant object (e.g. http://myvariant.info/v1/variant/rs12190874). Note that it supports dot notation as well, e.g., you can pass "dbsnp.rsid". If "fields=all", all available fields will be returned. Default: "all".<br>dbsnp |
| dotfield<br>boolean<br>*(query)* | control the format of the returned variant object. If "true" or "1", all fields will be collapsed into a single level deep structure (all nested objects will be a single level deep, using dotfield notation to signify the nested structure) Default: false.<br>-- |
| callback | |


Response body

```
{
  "_id": "chr6:g.152708291G>A",
  "_version": 2,
  "dbsnp": {
    "_license": "http://bit.ly/2AqoLOc",
    "alleles": [
      {
        "allele": "G",
        "freq": {
          "exac": 1,
          "gnomad": 1,
          "gnomad_exomes": 1,
          "topmed": 1
        }
      },
      {
        "allele": "A",
        "freq": {
          "exac": 0,
          "gnomad": 0,
```

Figure 2.3.2 result from MyVariant.info. (a) an example of how we query in MyVariant.info services. (b) part of the response body.

There are a lot of advantages to MyVariant.info. First, it integrates data from both large and specific databases. With large databases, we could know most information about a variant and with specific sources, we could access specialized fields. Also, a successful website service needs to be stable, scalable, in a good performance and MyVariant.info fits it well. It uses an indexing engine that provides not only various query syntax but also superior query performance. Moreover, MyVariant.info automates the updates weekly so that information will be up to date. (Xin et al. 2016)

There are 25 different databases contained by MyVariant.info, covering 1402 data fields and there is a weekly update to make sure users could access the latest information from these databases. There is an Application programming interface (API) for Python that enables programmatic access to the databases mentioned above and helps users annotate variants. MyVariant.info’s API offers many parameters with different query functions and many database categories and it is very flexible so individuals could query based on their own annotation needs. The annotation means the features of variants, such as physical position, chemical features, biological functions, etc.

But MyVariant.info has some limitations. For example, query large files can’t be executed on website servers and non-expert still have some barriers to query annotations. To address these problems, I developed a Python package to satisfy both experts and general individuals.

Chapter 3

# EVAT: PYTHON MODULE AND COMMAND-LINE INTERFACE

## *3.1 Overview*

The first aim of EVAT is to help people who have some bioinformatics or programming expertise in extracting variant-related information. With the API offered by EVAT, people with programming skills could retrieve variant information via Python.

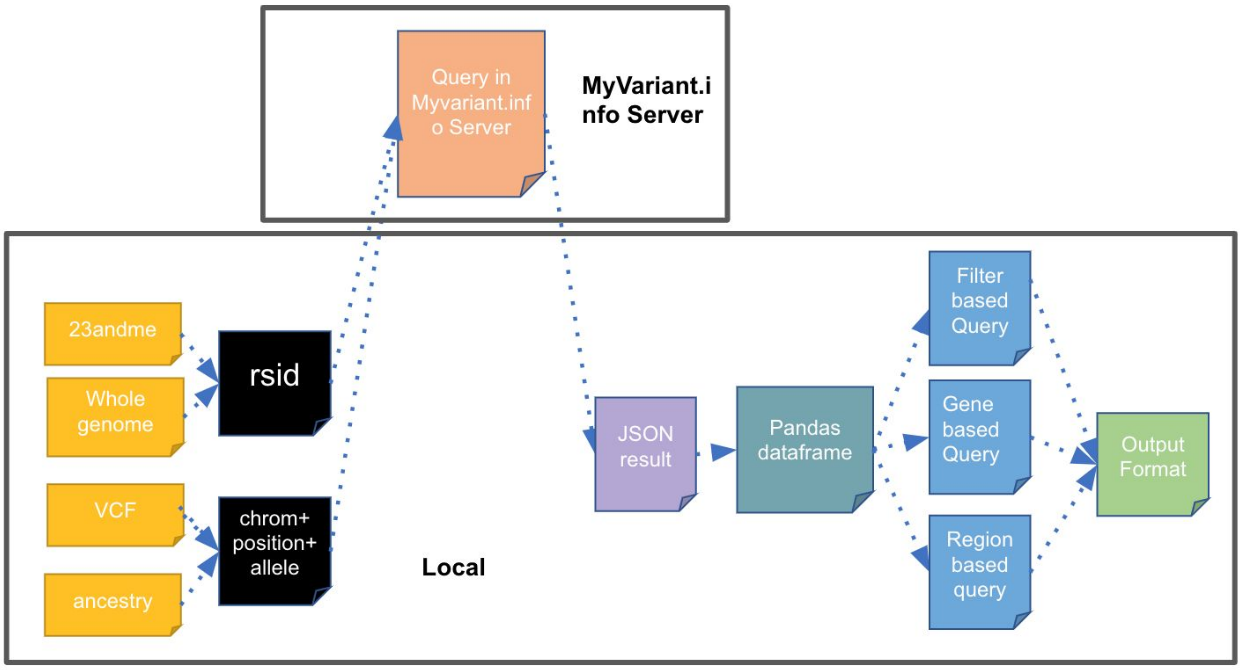


Figure3.1 the pipeline of the EVAT

Figure3.1 is the pipeline of the EVAT. The first step is the users upload a file which could be 23andme, whole genome, vcf or ancestry. The second step is that EVAT will convert those file formats to standard keys(rsid and chromosome + position + allele) which could be recognized by MyVariant.info API. Then EVAT will send those keys to MyVarant.info and get the result with JSON format. Because the JSON file format is not easy to read, EVAT will convert JSON result to pandas dataframe and then offers three query functions to help users get certain variant annotations. Finally, EVAT could output the file with VCF or CSV format.

### ***3.2 Dependencies in EVAT:***

PyLiftover: PyLiftover is a library for quick and easy conversion of genomic (point) coordinates between different assemblies. In this project, it was used to convert the reference genome version, e.g., from hg38 to hg19 because MyVariant.info is built on the hg19 assembly but the input file might be hg38. (pyliftover )

Pandas: *pandas* is an open-source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. In this project, pandas was used to manipulate data frames so that downstream operations could be done more easily. (pandas - Python Data Analysis Library )

## 3.3 Input Files

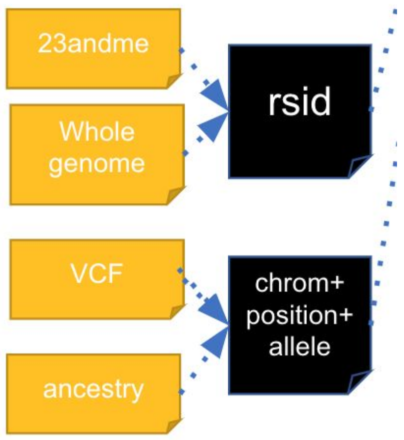


Figure3.3 loading files and converting to standard key

Different users may have different files. Users might have ordered a 23andMe kit and downloaded the results file or a biologist with a whole genome file in a standard format from a patient with cancer. However, none of these files can currently be uploaded to MyVariant.info directly because of differences in file formats. Therefore, the package offers a function that can parse the 23andMe, vcf, Ancestry and whole genome file to the format that could be uploaded to MyVariant.info to retrieve information.

After getting the input files, the following operations were done to prepare them for a MyVariant.info query. The first step is to convert the input file into a standard format based on the parameters. The standard format means rsid, such as rs12190874, or chromosome + position + genotype + mutation, for example: “char1:3415135A>G. Based on the different file types (read from parameter), the input file is interpreted by the corresponding parser. Currently, we have implemented four processors to deal with four file types (VCF, 23andMe, AncestryDNA, whole genome). Every module reads line-based information and breaks a line based on the “Tab” (“\t”) character or other delimiters. For example, in a VCF file, the module extracts chromosome, position and reference/alternative alleles. Then, the module put them together to form a standard query key and pass it to the next module.  In the future, we anticipate the use of new file formats and expect to add more processors in our package. The modular design of this part allows us to add processors more easily in the future.

### ***3.4 Interact MyVariant.info***

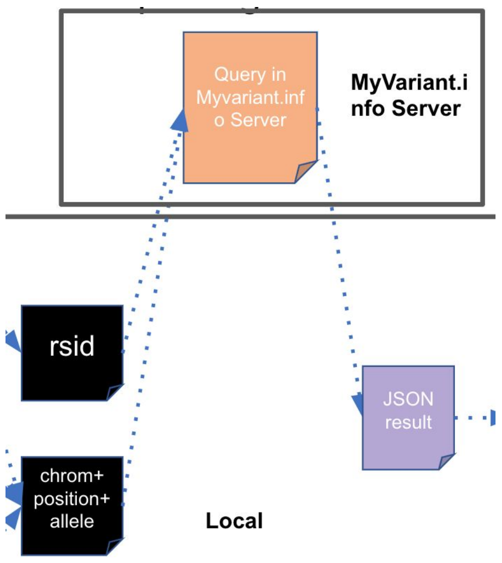


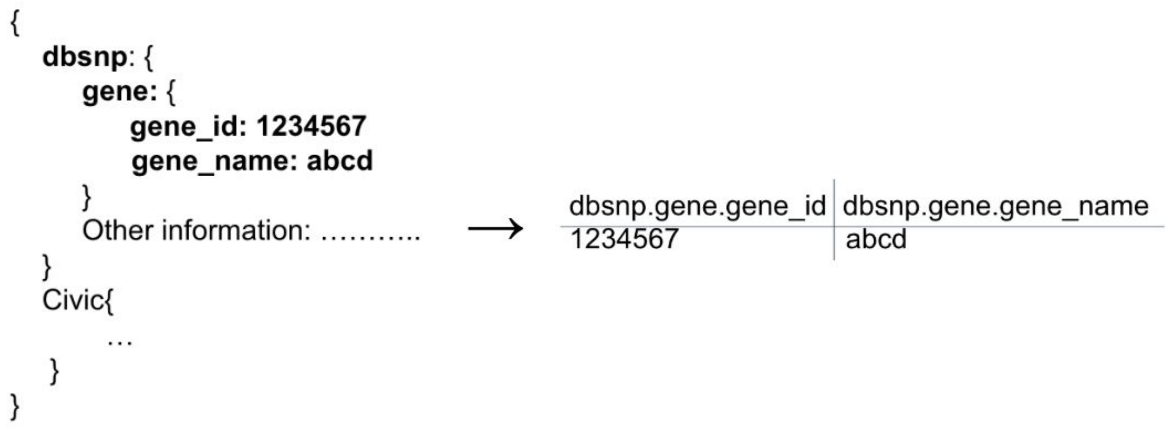
Figure 3.4 interacting with MyVariant.info: send keys to MyVairnt.info and get JSON results.

As MyVariant.info’s Python API requires a set of query parameters, which include the data file, file type, reference genome version, and the query field. Therefore, EVAT package’s query function is based on MyVariant.info query requirement. The package uses the standard format to form a query key and use the MyVariant.info query API to retrieve information.

Users have their own fields of interest and EVAT package needs to perform many functions corresponding to their needs. Thus, the package offers the user a basic query function that is similar to MyVariant.info as well as some new functions that have many new features compared to the query functions of MyVariant.info. The basic functions have many parameters and the first one is a field parameter. The “field” parameter allows users to use field names to query certain databases, such as “dbsnp” or “cadd”. Also, users could input many fields at one time to make a combined query. For example, [“dbsnp, “cadd.1000p”] could also be identified by the system.

In the next paragraph, I will state how EVAT converts results from MyVariant.info to human readable tables. First, I will introduce two data structures: Python’s “Dictionary” and the JSON format. Dictionary is a data structure in Python that is similar to HashMap and nested Dictionary is similar to nested HashMap. The nested HashMap here refers to the situation that the values of the HashMap are also a HashMap. The Hashmap is a collection of key value pairs based on the harsh function and it enables people to query keys with constant time. The JSON file is automatically converted to a nested dictionary when returned from MyVariant.nifo and the key is first level field name, dbsnp for example, and the value of them is also a dictionary, dbsnp.gene for example, and fields in dbsnp.gene is also a dictionary.

For example, Figure 3 shows a sample of JSON data. There are two elements in the first level dictionary: dbsnp and Civic. However, dbsnp is also a dictionary and gene and other information are the elements in this dictionary, which is an element of a  dictionary of a dictionary and we define this as a second level dictionary element. “gene” is also a dictionary and we need to call expand function again. To generate an output, we have to expand to human reader-friendly formats which are shown in the right-side figure. Every nested dictionary is an expanded before further operation.



```
{
   dbsnp: {
      gene: {
          gene_id: 1234567
          gene_name: abcd
      }
      Other information: ………..
   }
   Civic{
        …
    }
}
```

| dbsnp.gene.gene_id | dbsnp.gene.gene_name |
|---|---|
| 1234567 | abcd |

Figure 3.4.1**:** Example of converting JSON to the pandas.dataframe with a built-in function to help other operations.

EVAT package is based on MyVariant.info API and the data is returned in JSON file format. However, JSON files are hard to read and understand data structure by users. . To do additional operations, the package needs to get rid of all nested dictionaries and convert it to another file format, and we used a pandas dataframe here. The method is for every element in the first level dictionary, if it is still a dictionary, we call expand function with this dictionary and use the name of this dictionary as input parameter like dbsnp. The expand function converts the element in a dictionary to an element in the outer list. Then for every expanded element, we merge the name of all levels and call expand function again if the element is still a dictionary. In the end, every element is expanded in standard format such as dbsnp.1000g.af and we could use them as pandas. dataframe.

## *3.5 EVAT Special Built-In Special Query Functions*

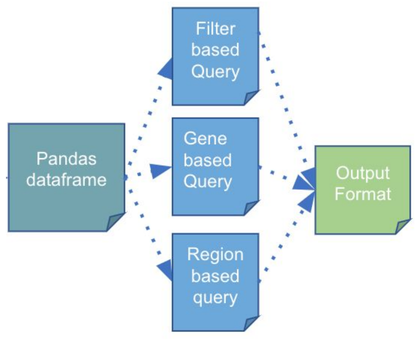


Figure 3.5 three query functions offered by EVAT.

EVAT offered gene-based annotation, region-based annotation, and filter-based annotation to fit the users' needs. Gene-based query subsets out variants belonging to certain genes. A variant is located on the chromosome and the location might belong to one or more genes. The module tries to find the gene at that location and tell users which gene is affected. When it comes to region-based annotation, it is a method to help users identify whether the variant is located in a certain region of the chromosome. (Li 2011) There are many specific regions on the genome, such as binding sites and conserved regions. Users could pick the regions they care about, and the program maps the region and the variants' locations. For example, users could use chromosome1:15000-chromosome1:300000 and the variants in this are identified. The last query funtion is a filter-based query, which is flexible. Filter-based queries accept a range and a field, and then the program will find variants that have the field in that range. For instance, the user could filt the variants that have frequency between 0.2 and 0.8 by the filter-based query. These three annotative methods are used in common tasks and EVAT offers complete functions to do the related analysis.

## *3.6 Output File Format*

EVAT returns two different output file formats: VCF and CSV. CSV stands for "comma-separated values", in which all data are separated by a comma. Within the data frame, the package adds a comma between the fields. Therefore, all fields are separated by commas and each variant corresponds to a line in the file.

To convert a dataframe into CSV file, the package first converts the dataframe into string, and then changes all tabs into a comma. For example, for elements in a line, this function read them one by one and convert them into a string and put them together to form the output file.

As for the VCF file, the package first writes the header block according to the latest VCF specifications. The version information is retained from the input and the head listed below is the VCF standard format. In VCF headers, chromosome, position, ID, reference, alternative type listed below are all queried based on the dbsnp database. The INFO is the information the user wants and in the INFO, the data is organized by "title: data, title2: data2….".

```
##fileformat=VCFv4.1

##fileDate=2019-04-16 21:50:19.076473

##version=hg19

##CHROM     POS    ID       REF     ALT     QUAL  FILTER          INFO
```

## *3.7 Time and Memory Requirements:*

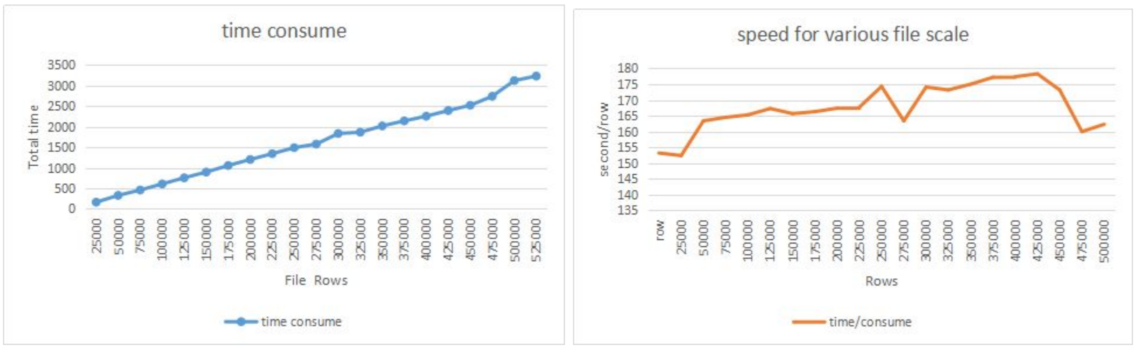


Figure 3.7.1: the x-axis is the number of variants. Left: Time consumed for certain row variants. Right: speed row_number we could annotate per second. Like every. The y-axis is the time consumed by every 25000 lines.

To test if our package could run both on a personal computer (Windows 10, ryzen5) and on the server (Linux CentOS 5), several tests were performed. We used a laptop to query all 500,000 variants with 16 gigabytes memory. About an average of 3400 seconds later, the laptop finishes the process of querying from MyVariant.info and works fine. On the server, we test with the same set and use a similar time consume. The result is similar: for every 25000 lines, the time cost is about 150 second. The result means that EVAT is efficient and stable when annotating large files. After querying all variants in the file, subsequent operations such as gene-based query would be quick, taking only a few minutes.

For experts in bioinformatics, time complexity and space needed are important. EVAT can parallelize the code  to reduce the time cost. Therefore, the time cost will decrease to two-fifth of the original time. When it comes to the local disk management, EVAT offers web-based queries so that users will not need to download extra files and our package can save disk space for them. However, there is a bottleneck in querying from MyVariant.info and it is hard to do the query quicker and this limitations will be discussed in Chapter6.

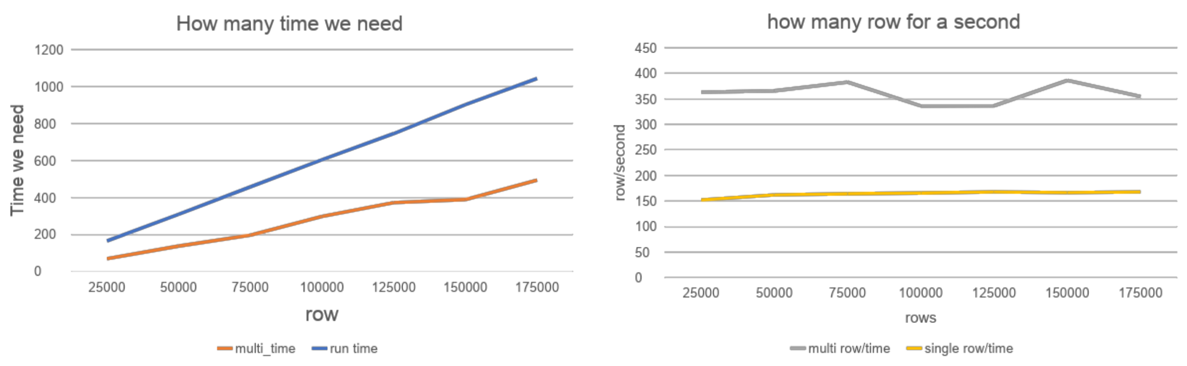


Figure 3.7.2: the x-axis is the number of variants. Top: y-axis is time consumed for certain row variants. Blue line is a program without parallel and the orange line is a program with parallel. Bottom: y-axis is the speed that we could annotate per second.

Chapter 4

# EVAT: Graphical User Interface

## *4.1 Introduction to Graphical User Interface*

EVAT offers users who do not have program expertise in retrieving and interpreting variant data an easy-to-use interface so that they could upload raw data without having to write code. The graphical user interface is common for most people who have experience accessing websites and apps, etc. The program uses certain functions to query and interpret the data in the file and output an annotated file to the user. The output file contains the genetic annotation information retrieved from MyVariant.info. It groups related information and presents it to users to help form meaningful conclusions. For example, it gathers the information in the same pathway so that if an unusually large number of variants show up, one can hypothesize that this pathway is disrupted. With the interface and interpreted output reports, people without informatics expertise could understand their data.

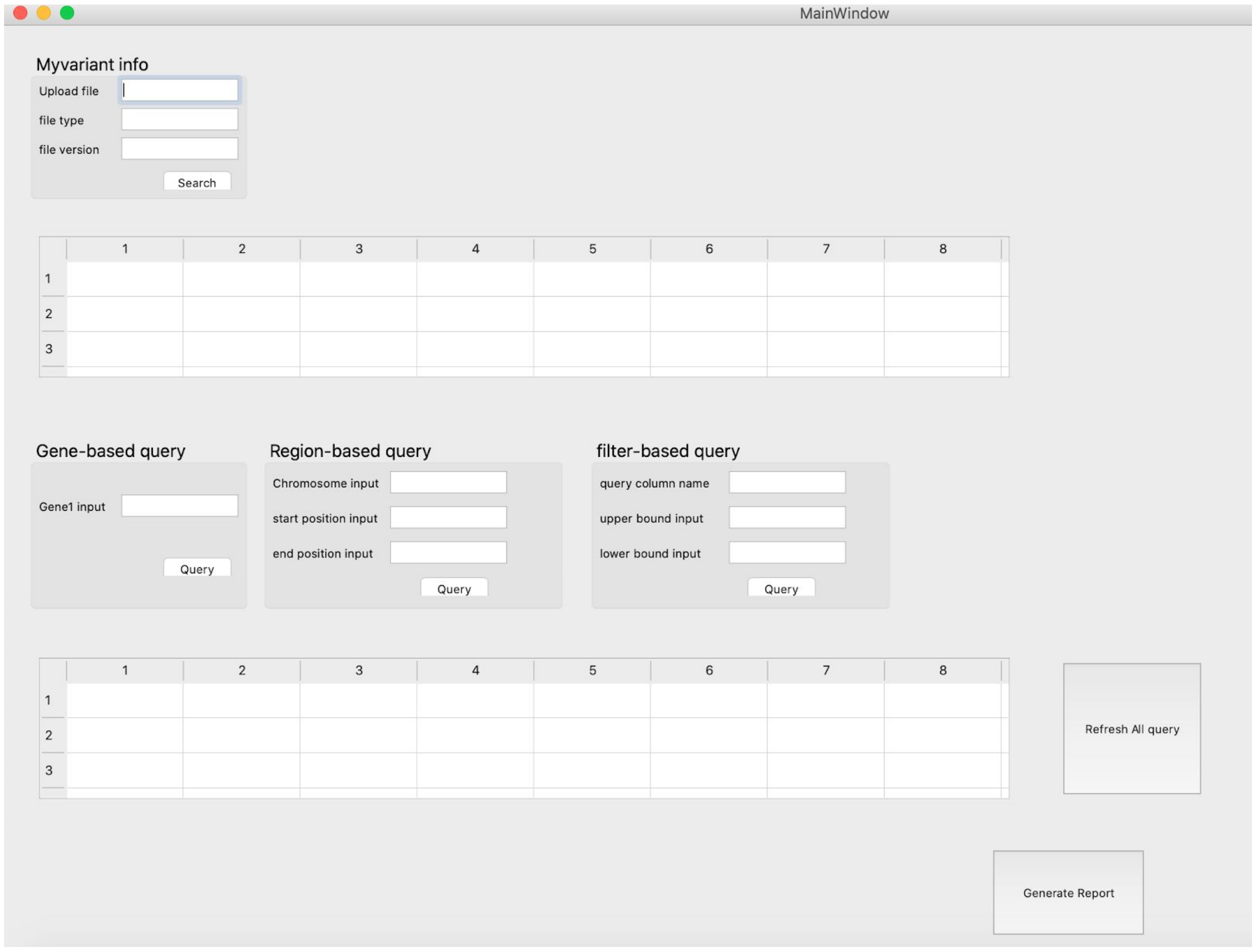


**Figure4.1.1: Illustration of User-Interface.** Simple user-interface with widget groups for uploading, loading table, querying functions and resulting table.

### *4.2 Graphical User Interface for EVAT*

To help users without a programming background, the Python package has a built-in graphical user interface(Figure4.1.1). With that graphical user interface, users could get their annotation by mouse and several simple input, like file name. Therefore, users could do queries without any coding skills.

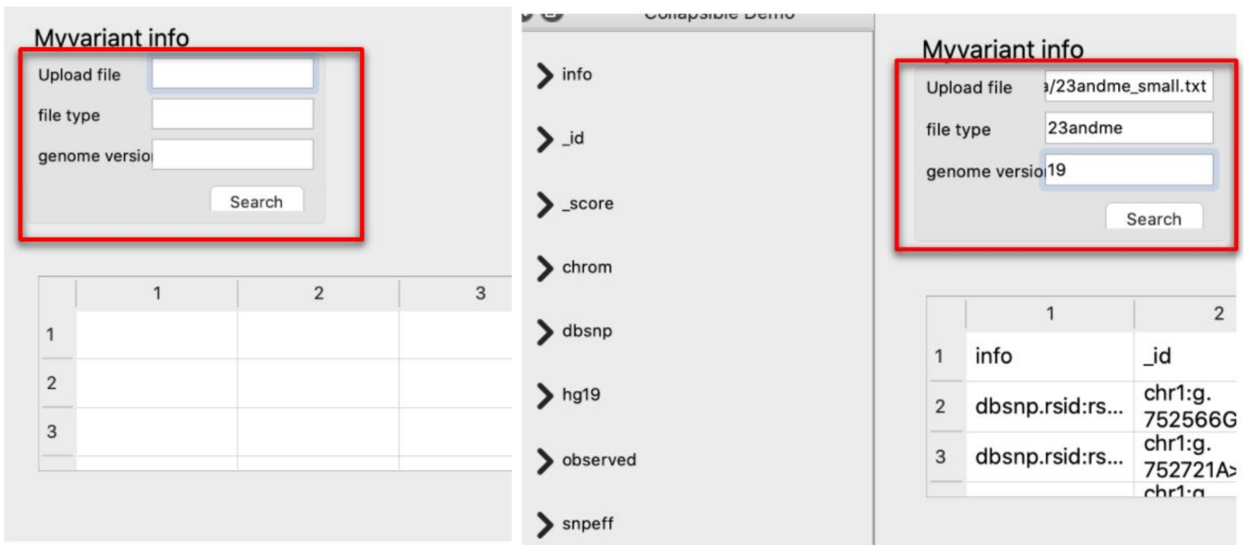
The graphical user interface is built with four main components. The first one is file upload group box, which has three textbox; upload file name, genome file type and reference genome version. Users need to upload files here to EVAT. Then, the collapsible box shows up on the left, each parent node is a database and each leaf node is a field of that database. On the right, there is a loading table that shows every annotation to the user. Below the table, there are the query functions. And also, users could input the parameter under the box and do the query to the column they selected. And after the query has finished, users could get their result shown in the result table.

For users who don't have programming skills and informatics background, the mouse is a more friendly input device than a keyboard. For most users, the mouse is a more common input device. With a terminal and keyboard, the user needs to know how to use the function and how to import the python functions. For example, "query(file, vcf, 19, conserved)" is a query function, but users need to understand the document and know the meaning of every corresponding parameter. The documentation could tell the meaning and usage of the functions and parameters but individuals without expertise will spend a lot of time doing that. With the graphical user interface, all they need to know are following the guide in Chapter 5. Also, they just need to input the parameters just like all other applications on the web.

## *4.3 Code Structure of the Front End*

I used PyQt5 to build the graphical user interface as it has many advantages. First, it is fast and stable. PyQt5 is a package that developed in 2016, but the Qt (which is the precursor of PyQt5) was developed 24 years ago and it gradually became one of the most popular packages for graphical user interfaces. PyQt5 is also an easy-to-use package. Moreover, pyQt5 could be used on all platforms like Mac, Windows, Linux, etc.

To build the graphical user interface, the package uses groups that contain many small widgets of a part. For example, the upload group contains Upload file prompt, upload file textbox, reference genome version hint, reference genome version textbox. All these widgets are wrapped into a group and could be moved together. Therefore, we have implemented these widgets as one thing to operate them at the same time. For example, when a collapsible box is shown and these widgets need to move together, we don’t have to move one by one but change the widget group coordinate instead. (Figure4.3) So the problem is how we build that widgets group. Generally, the widgets group is allocated a location first based on the design of the graphical user interface. Then, the widget is allocated a relative location. For instance in figure 4.3, the upload  function group box is located at position(x: 20, y: 20) and the upload file box is at (x: 10, y: 30). The real absolute location of this upload file box is at (30, 50). However, because it is a widget in the upload  function group box, we assign it as (x: 10, y: 30) and this widget will show up on the (x: 10, y: 30) beginning from the top left corner of the widget group box. And if we move the location of the upload file group box to (70, 20) when the collapsible menu shows up. All locations of widgets will shift 50 right on the x axis. By that feature, all widgets groups are generated and assigned that way. (Figure 4.3)



Upload file: a/23andme_small.txt
file type: 23andme
genome versio: 19

| | 1 | 2 |
|---|---|---|
| 1 | info | _id |
| 2 | dbsnp.rsid:rs... | chr1:g. 752566G |
| 3 | dbsnp.rsid:rs... | chr1:g. 752721A> |

**Figure 4.3:** left is a figure that shows the upload file widget box in red. The red widget box in the right figure shifts 50 units right and all widgets in it do the same shift.

To build the collapsible box, there is a unique collapsible box function that defines a collapsible box, which contains an arrow button, a collection of leaf nodes and the logic after click. After the user clicks on the arrow, the arrow changes to “down” if it is “right” now. At the same time, the function also calculates the extra space of the collapsible box based on the leaf node numbers. When clicking the arrow when it is “down”, the arrow goes back to “right”. And all sub menus and nodes disappear and return the original status.

Chapter 5

# USER GUIDE FOR EVAT

## *5.1 Python package*

We built documentation and tutorials to help users navigate the graphical user interface. An early version of the step-by-step guide is presented below:

Check if the Python is already downloaded on your computer. For Mac users, Python is a built-in function, but for other operating systems, users need to go to https://www.Python.org/downloads/ to download the Python file and follow the https://www.Python.org/about/gettingstarted/ guide.

Open terminal. Type “pip install this package” and then “Type Python3 package Interface” and the graphical user interface will show up.

## *5.2 Python Package Guide*

To use EVAT, users could use the Python package in EVAT. For example, users could build the annotation class by the EVAT python package and do the queries they need. Here is a demo of python. The first step is to build a query object with the input file name, output file name, genetic type of input file, genetic reference version, fields of query, begin and end lines. The second step is to do the query and the next three lines are different queries.

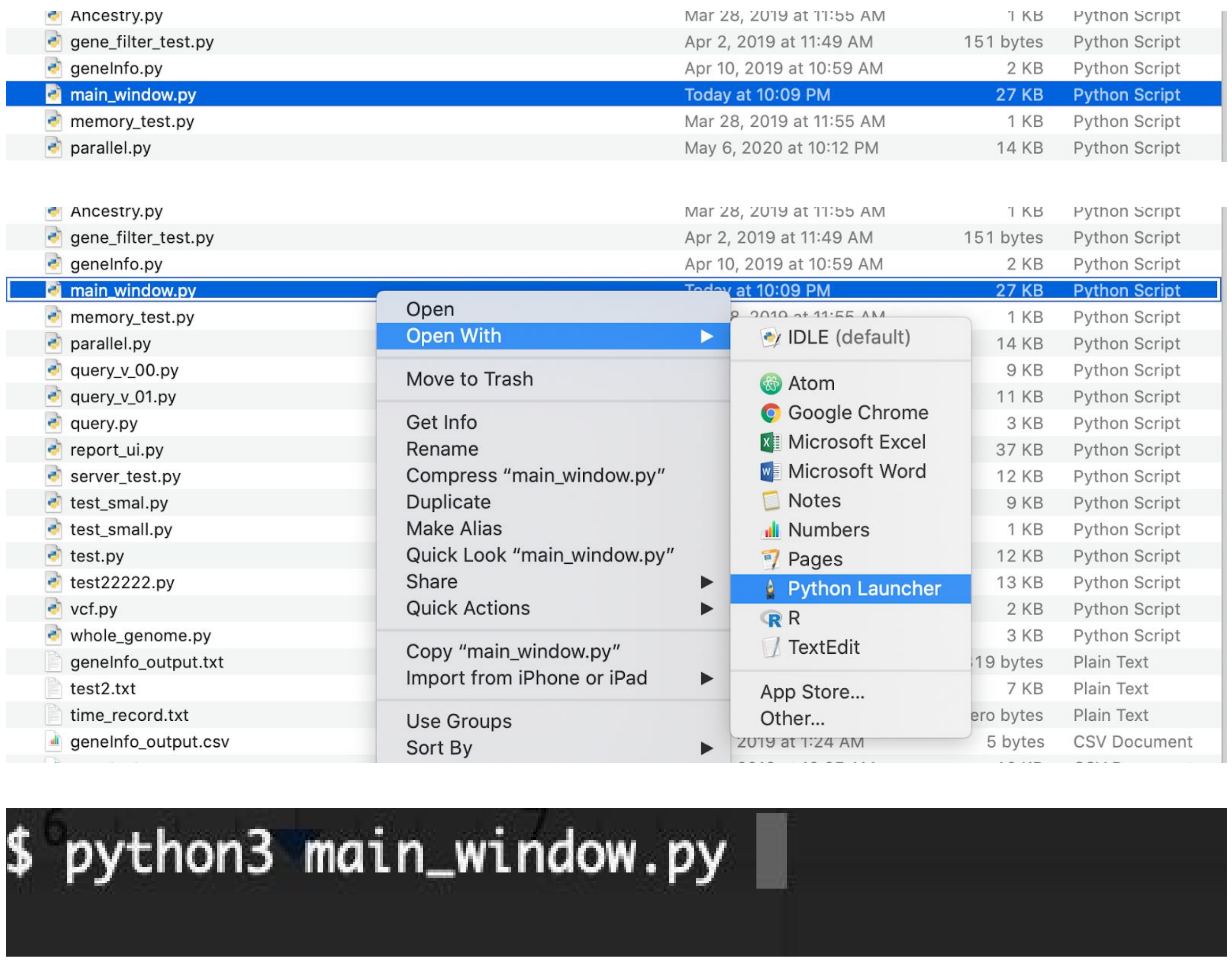
```
demo = query('../Data/23andme_small.txt','test.csv','23andme','csv', 19,'all',1,1000)

demo.genequery()

demo.geneBasedQuery('OR4F5-LOC729737')

demo.filterBasedQuery('cadd.1000g.af', 0.6, 0.2)

demo.regionBasedQuery(1, 1000, 800000)
```

Figure 5.2.1 demo of a query from EVAT.

### *5.3 Graphical User Interface Guide*

For users without a programming background, a graphical user interface is offered. A basic query involves seven different steps:

Step1: Launching the interface. To launch the graphical user interface,  users can find the Python file(Figure 5.3.1.a) and open it with Python Launcher(Figure 5.3.1.b), or use the terminal with command “Python3 main_window.py”(Figure 5.3.1.c) to open the front end.



|  |  |  |  |
|---|---|---|---|
| Ancestry.py | Mar 28, 2019 at 11:55 AM | 1 KB | Python Script |
| gene_filter_test.py | Apr 2, 2019 at 11:49 AM | 151 bytes | Python Script |
| geneInfo.py | Apr 10, 2019 at 10:59 AM | 2 KB | Python Script |
| main_window.py | Today at 10:09 PM | 27 KB | Python Script |
| memory_test.py | Mar 28, 2019 at 11:55 AM | 1 KB | Python Script |
| parallel.py | May 6, 2020 at 10:12 PM | 14 KB | Python Script |

|  |  |  |  |
|---|---|---|---|
| Ancestry.py | Mar 28, 2019 at 11:55 AM | 1 KB | Python Script |
| gene_filter_test.py | Apr 2, 2019 at 11:49 AM | 151 bytes | Python Script |
| geneInfo.py | Apr 10, 2019 at 10:59 AM | 2 KB | Python Script |
| main_window.py | Today at 10:09 PM | 27 KB | Python Script |
| memory_test.py |  | 1 KB | Python Script |
| parallel.py |  | 14 KB | Python Script |
| query_v_00.py |  | 9 KB | Python Script |
| query_v_01.py |  | 11 KB | Python Script |
| query.py |  | 3 KB | Python Script |
| report_ui.py |  | 37 KB | Python Script |
| server_test.py |  | 12 KB | Python Script |
| test_smal.py |  | 9 KB | Python Script |
| test_small.py |  | 1 KB | Python Script |
| test.py |  | 12 KB | Python Script |
| test22222.py |  | 13 KB | Python Script |
| vcf.py |  | 2 KB | Python Script |
| whole_genome.py |  | 3 KB | Python Script |
| geneInfo_output.txt |  | 19 bytes | Plain Text |
| test2.txt |  | 7 KB | Plain Text |
| time_record.txt |  | ero bytes | Plain Text |
| geneInfo_output.csv | 2019 at 1:24 AM | 5 bytes | CSV Document |

```
$ python3 main_window.py
```

Figure 5.3.1: Steps to open graphical user interface.(a)find main_window.py. (b)open with Python Launcher. (c) open with a terminal.

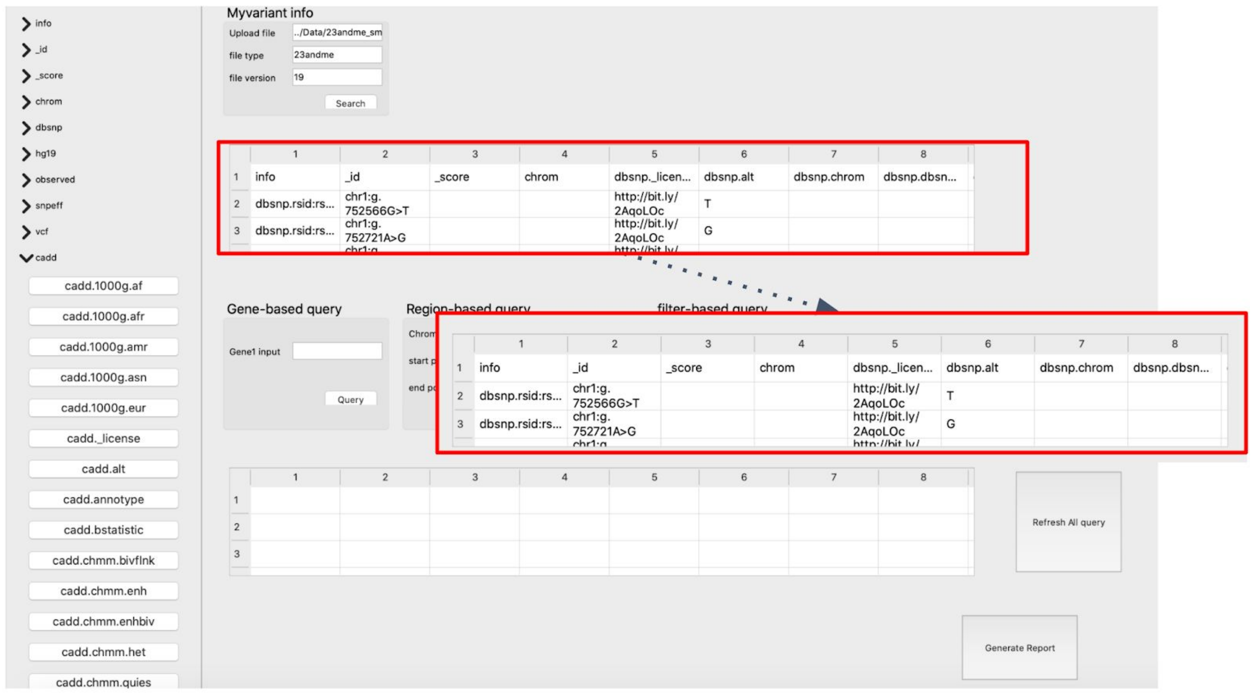
Step2: File upload

After opening the graphical user interface, users will see the UI listed below. On the top left, the graphical user interface offers a window that could upload files. Users need to input file name, file type and reference genome version. For example, the file name is 23andme_small.txt; file type is 23andme, reference genome version is 19.

Figure 5.3.2: Steps2, file upload window.

Step3:

After the file is uploaded, the loading table(top table) will show annotations of every variant. Every row is a variant and the column is the annotation field. All the columns will show on the left and it is a collapsible menu. Users could use the collapsible menu to browse the field names.



Myvariant info

Upload file ../Data/23andme_sm

file type 23andme

file version 19

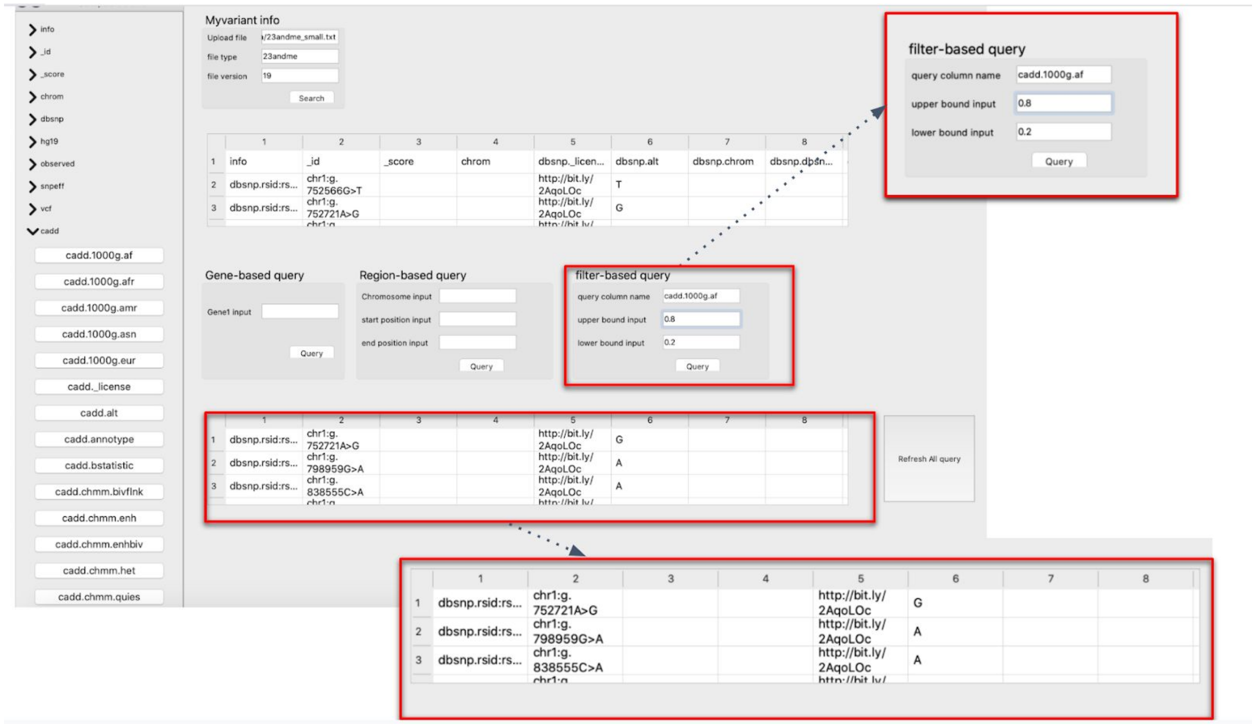
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | info | _id | _score | chrom | dbsnp._licen... | dbsnp.alt | dbsnp.chrom | dbsnp.dbsn... |
| 2 | dbsnp.rsid:rs... | chr1:g. 752566G>T | | | http://bit.ly/ 2AqoLOc | T | | |
| 3 | dbsnp.rsid:rs... | chr1:g. 752721A>G | | | http://bit.ly/ 2AqoLOc | G | | |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |

Figure 5.3.3: Steps3, loading table.

Step4:

After that, the users could do three kinds of queries as shown in the middle of the graphical user interface. Every query type has unique query parameters. Filter-based query, as example, has three parameters: query column name that tells the tool which field you are interested in; upper bound input and lower bound input tell the tool the range users need. The result after the query is shown in the result table below.



Myvariant info

| | |
|---|---|
| Upload file | ı/23andme_small.txt |
| file type | 23andme |
| file version | 19 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | info | _id | _score | chrom | dbsnp._licen... | dbsnp.alt | dbsnp.chrom | dbsnp.dbsn... |
| 2 | dbsnp.rsid:rs... | chr1:g. 752566G>T | | | http://bit.ly/ 2AqoLOc | T | | |
| 3 | dbsnp.rsid:rs... | chr1:g. 752721A>G | | | http://bit.ly/ 2AqoLOc | G | | |

Gene-based query

| | |
|---|---|
| Gene1 input | |

Region-based query

| | |
|---|---|
| Chromosome input | |
| start position input | |
| end position input | |

filter-based query

| | |
|---|---|
| query column name | cadd.1000g.af |
| upper bound input | 0.8 |
| lower bound input | 0.2 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | dbsnp.rsid:rs... | chr1:g. 752721A>G | | | http://bit.ly/ 2AqoLOc | G | | |
| 2 | dbsnp.rsid:rs... | chr1:g. 798959G>A | | | http://bit.ly/ 2AqoLOc | A | | |
| 3 | dbsnp.rsid:rs... | chr1:g. 838555C>A | | | http://bit.ly/ 2AqoLOc | A | | |

filter-based query

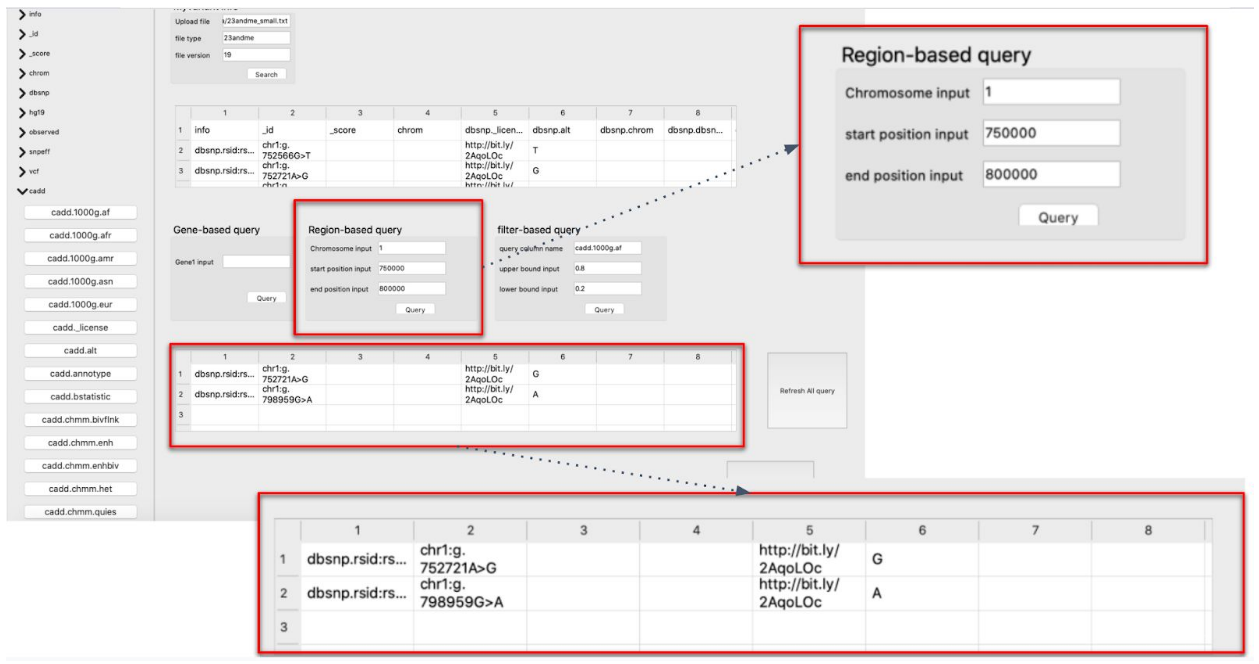
| | |
|---|---|
| query column name | cadd.1000g.af |
| upper bound input | 0.8 |
| lower bound input | 0.2 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | dbsnp.rsid:rs... | chr1:g. 752721A>G | | | http://bit.ly/ 2AqoLOc | G | | |
| 2 | dbsnp.rsid:rs... | chr1:g. 798959G>A | | | http://bit.ly/ 2AqoLOc | A | | |
| 3 | dbsnp.rsid:rs... | chr1:g. 838555C>A | | | http://bit.ly/ 2AqoLOc | A | | |

Figure 5.3.4: Steps4, three functions and result table.

Step5:

Also, Another query could be done after the query above and the result will be the intersection of these two queries. Here we use region based query as an example. There are three parameters in region based query: first one is which chromosome you want and the second and third are the start position and end position of the query. After the query below, There are only two rows in the result table.



Upload file: /23andme_small.txt
file type: 23andme
file version: 19

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | info | _id | _score | chrom | dbsnp._licen... | dbsnp.alt | dbsnp.chrom | dbsnp.dbsn... |
| 2 | dbsnp.rsid:rs... | chr1:g. 752566G>T | | | http://bit.ly/ 2AqoLOc | T | | |
| 3 | dbsnp.rsid:rs... | chr1:g. 752721A>G | | | http://bit.ly/ 2AqoLOc | G | | |

Region-based query
Chromosome input: 1
start position input: 750000
end position input: 800000
Query

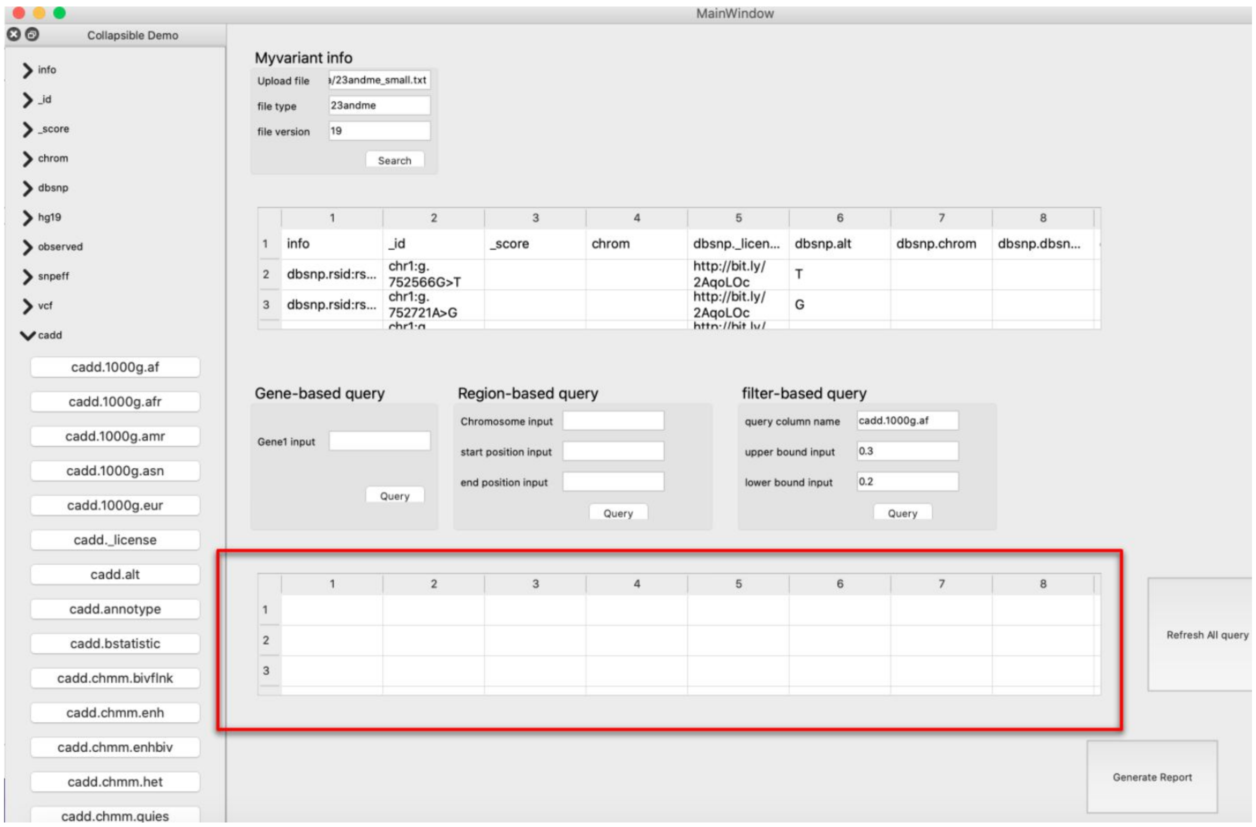
filter-based query
query column name: cadd.1000g.af
upper bound input: 0.8
lower bound input: 0.2
Query

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | dbsnp.rsid:rs... | chr1:g. 752721A>G | | | http://bit.ly/ 2AqoLOc | G | | |
| 2 | dbsnp.rsid:rs... | chr1:g. 798959G>A | | | http://bit.ly/ 2AqoLOc | A | | |
| 3 | | | | | | | | |

Figure 5.3.5: Steps5, do another query after Step4 and the intersection result is shown on the result table.

Step 6:

Also, The result table could be cleared. Users could use Refresh all query functions and do the query again. This function is used when users want to do another query or delete the query now.



Myvariant info

| | |
|---|---|
| Upload file | ı/23andme_small.txt |
| file type | 23andme |
| file version | 19 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | info | _id | _score | chrom | dbsnp._licen... | dbsnp.alt | dbsnp.chrom | dbsnp.dbsn... |
| 2 | dbsnp.rsid:rs... | chr1:g. 752566G>T | | | http://bit.ly/ 2AqoLOc | T | | |
| 3 | dbsnp.rsid:rs... | chr1:g. 752721A>G | | | http://bit.ly/ 2AqoLOc | G | | |

Gene-based query

| | |
|---|---|
| Gene1 input | |

Region-based query

| | |
|---|---|
| Chromosome input | |
| start position input | |
| end position input | |

filter-based query

| | |
|---|---|
| query column name | cadd.1000g.af |
| upper bound input | 0.3 |
| lower bound input | 0.2 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |

Figure 5.3.6: clear all query buttons that could clear all tables and do that again.

Step 7:

The Generate report button could output the result in the result table to a csv file.

### *5.4 Case Study*

In this section, I describe a case study using real PGP(Personal Genome Project) data and EVAT. The PGP data includes a 23andme data file named huED0F40 which can be downloaded from PGP Data and the health report of a participant could be found in the PGP Genome Report. The huED0F40 file is the 23andme file which contains 576532 variants of the participant. The health report includes the current disorder or disease of the participant and the potential risk.

The participant was suffering loss of hearing and diagnosed with Asymmetrical Hearing

Loss caused by genetic mutations. Therefore, he wanted to know more about his mutations and decided to do a genetic test. Then he got his 23andme file but he knew nothing about those variants as there are only some rsid there. So he wanted some more information and used EVAT to retrieve more information about his genetic variants and got all fields annotation of these variants. Then he learned about some gene names related to Asymmetrical Hearing Loss by reading papers, which is GJB2, SLC26A4, MYO15A, OTOF, CDH23, and TMC1. Let's use GJB2 as an example. By using EVAT, and its gene-base querying ability, he found 9 valid variants related to those genes. But not all of these variants were equally important and some of them were more common and well researched. Therefore, he used filter based query function in EVAT to get variants that have dbnsfp.mutpred.score higher than 0.75, (dbnsfp.mutpred.score higher than means this variant has a high risk of causing a disease) and had gnomad_exome.af.af <= 0.01 (variants that located on genes where one or more identified high-impact SNVs have AF > 0.1% in gnomAD) (Whiffin et al. 2020). After the query, he got one variant that is rs28931595. By the other field in the output file, he learned more information about that variant. Thus, the EVAT tool can help such users find more information about their most relevant variants

Chapter 6

# Discussion

### *6.1 EVAT*

EVAT could offer variant annotation information to individuals with different backgrounds. Everyone could get variants annotations from the Python package or from the graphical user interface. The package has many features to help users retrieve their variant annotation by their own way. In the backend structure, EVAT offers the four different input file interpreters, including 23andMe, vcf, Ancestor and the whole genome.  Also, the query function offers parameters so that users could custom their query statement to MyVairant.info. After that, EVAT offers 3 high-level query functions: Gene-based funcion, filter-based function, region-based function to do the query for result from MyVariant.info. Then, the output file format could be chosen from vcf or csv.

Furthermore, EVAT includes a graphical  graphical user interface for users without a programming background. With the mouse click and the input to the textbox, Users could get annotation data from MyVariant.info, browse the data by table and browse the columns and fields by collapsible box. Then, the user could use the 3 functions to query their data and the output file they need.

## *6.2 Advantages*

For users with different backgrounds, EVAT could offer different variant annotation methods. For users with programming skills, EVAT offers an API for users to control every step of query input parameters, query functions, input and output file format. There are functions in the Python package that could be imported by users.

For users without a programming background, EVAT offers the user an graphical graphical user interface. With that graphical user interface, users could also control every step of query input parameters, query functions, input and output file format or ignore everything and output everything. However, the steps could not be imported  for other usage. For example, users with programming experience could import EVAT functions and use them in their code, but with user interface you can only do what graphical user interface can do.

EVAt queries information from the MyVairant.info, which could offer a large amount of information. EVAT is not only friendly to all users but also offers various functions that optimize query experience. Also, a high-performance API that allows real-time queries on the Python package is provided.

## *6.3 Limitations and Disadvantages*

EVAT still has some limitations and Disadvantages.

Firstly, at certain situations, users may find the annotation is wrong or they have new features of the variant and want to contribute to the variant annotation. However, they could not add them through EVAT. What they could do is email the related databases like Dbsnp or CGI to add their research result to the whole database.

Secondly, the query in EVAT is based on the Internet and the speed of query can’t be too quick. A real variant file without annotation is about 500,000 lines and 10MB - 100MB. And it takes about 6 hours to do all the queries. Although queries may not cover all fields and the file might be part of the complete file, the efficiency of the whole process is not good enough. The bottleneck is to query from the MyVarint.info and this is hard to improve. Unless MyVariant.info changes the APIs that allow many lines to query at the same time.

Thirdly, the graphical user interface is not fully developed. There are still some limitations. For example, when the input file is larger than one million rows, the loading speed becomes slow dramatically. Also, the Python is required when launching the graphical user interface and the users need to download Python to use EVAT.

Furthermore, EVAT could only query data in human species. The annotation data in EVAT is from MyVariant.info and MyVairnat.info doesn't cover annotation further than human. Therefore there can’t be annotation further than that. However, connecting to other resources means a complete pipeline: EVAT needs to load more data types, interact with other API, interpret returned files and do more query functions. And these may be done in the future.

Last, the annotations are focused on SNV. The other variant type is not as frequent as SNV, but they play important roles in the biology process. However, the annotation based on that part is not completed and can not be annotated by EVAT.

## *6.4 Future Plan*

The package will offer an analysis function to users to visualize the result of the query. For example, the diseases related variants will be compared to other people and databases. Based on specific variants' occurrence frequency, the disease probability will be calculated.

Also, the visualization function needs to be added into the package. After the data are analyzed, the system could visualize the data into the figures. For example, the genetic variants frequency data will be shown in volcano plots to show the comparison of the variant and the general frequency.

Graphical user interface also needs some more features. For example, when the user selects a field or column, the interface should throw a window to notify the user what the column means as the column name is confused sometimes. For example, users may feel confused about the field ‘Cadd.1000g.af’, which means the Cadd database, 1000 genome project, african variant sequence.

In the future, the graphical user interface will contain a button that could generate human-readable reports for users. In this report, the ratio of disease variants that are positive and variants that are negative are offered.  And there is a description part here that the disease and related variants are described.

Also, we will compare the mutations with the general population so that the user knows if this mutation is rare or common. Finally, we will map the variant to the pathway and share the pathway that has many variants to users. If a pathway is enriched in disease-associated mutations, it is flagged as being noteworthy.

Finally, a chromosomal map will be generated by the package. The mapped variants show the overview of variants distribution for users. With this figure, users could have an overview of the variants they have and also know the locations.

Chapter 7:

# Summary

In this thesis, I built a Python package to help users  know what the variant is about by retrieving data from different databases and offering query functions. Users could be individuals with little or extensive programming knowledge and EVAT will help all of them. For users with programming skills, EVAT offers an API for users to control every step of query input parameters, query functions, input and output file format. For users without a programming background, EVAT offers the user an graphical  graphical user interface and this feature makes EVAT easy to use.

Also, EVAT has huge potential as EVAT is not a tool that will only be used in the bioinformatics field. In the future, EVAT can offer variant annotation information with genetic test participants with fancy visualizations and easy-to-understand reports by one click on the graphical user interface. By that way, EVAT may be able to put into market and benefit a lot of people.

## REFERENCES

Arnold, Matthias, Johannes Raffler, Arne Pfeufer, Karsten Suhre, and Gabi Kastenmüller. 2015. "SNiPA: An Interactive, Genetic Variant-Centered Annotation Browser." *Bioinformatics* 31 (8): 1334–36.

Bowne, Sara J., Lori S. Sullivan, Susan H. Blanton, Constance L. Cepko, Seth Blackshaw, David G. Birch, Dianna Hughbanks-Wheaton, John R. Heckenlively, and Stephen P. Daiger. 2002. "Mutations in the Inosine Monophosphate Dehydrogenase 1 Gene (IMPDH1) Cause the RP10 Form of Autosomal Dominant Retinitis Pigmentosa." *Human Molecular*

*Genetics* 11 (5): 559–68.

Chung, Jade C. S., Jennifer Becq, Louise Fraser, Ole Schulz-Trieglaff, Nicholas J. Bond, Juliet Foweraker, Kenneth D. Bruce, Geoffrey P. Smith, and Martin Welch. 2012. “Genomic Variation among Contemporary Pseudomonas Aeruginosa Isolates from Chronically Infected Cystic Fibrosis Patients.” *Journal of Bacteriology* 194 (18): 4857–66.

Cingolani, Pablo, Adrian Platts, Le Lily Wang, Melissa Coon, Tung Nguyen, Luan Wang, Susan J. Land, Xiangyi Lu, and Douglas M. Ruden. 2012. “A Program for Annotating and Predicting the Effects of Single Nucleotide Polymorphisms, SnpEff: SNPs in the Genome of Drosophila Melanogaster Strain w1118; Iso-2; Iso-3.” *Fly* 6 (2): 80–92.

“Definition of Variant - NCI Dictionary of Cancer Terms.” 2012. National Cancer Institute. July 20, 2012. https://www.cancer.gov/publications/dictionaries/genetics-dictionary/def/genetic-variant.

Douville, Christopher, Hannah Carter, Rick Kim, Noushin Niknafs, Mark Diekhans, Peter D. Stenson, David N. Cooper, Michael Ryan, and Rachel Karchin. 2013. “CRAVAT: Cancer-Related Analysis of Variants Toolkit.” *Bioinformatics* 29 (5): 647–48.

Landrum, Melissa J., Jennifer M. Lee, Mark Benson, Garth Brown, Chen Chao, Shanmuga Chitipiralla, Baoshan Gu, et al. 2016. “ClinVar: Public Archive of Interpretations of Clinically Relevant Variants.” *Nucleic Acids Research* 44 (D1): D862–68.

Li, Heng. 2011. “Tabix: Fast Retrieval of Sequence Features from Generic TAB-Delimited Files.” *Bioinformatics* 27 (5): 718–19.

Lin, Maoxuan, Sarah Whitmire, Jing Chen, Alvin Farrel, Xinghua Shi, and Jun-Tao Guo. 2017. “Effects of Short Indels on Protein Structure and Function in Human Genomes.” *Scientific Reports* 7 (1): 9313.

McLaren, William, Laurent Gil, Sarah E. Hunt, Harpreet Singh Riat, Graham R. S. Ritchie, Anja Thormann, Paul Flicek, and Fiona Cunningham. 2016. “The Ensembl Variant Effect Predictor.” *Genome Biology* 17 (1): 122.

“Pandas Documentation — Pandas 1.0.3 Documentation.” n.d. Accessed June 3, 2020. https://pandas.pydata.org/pandas-docs/stable/index.html.

Seeb, J. E., G. Carvalho, L. Hauser, K. Naish, S. Roberts, and L. W. Seeb. 2011. “Single-Nucleotide Polymorphism (SNP) Discovery and Applications of SNP Genotyping in Nonmodel Organisms.” *Molecular Ecology Resources* 11 Suppl 1 (March): 1–8.

Sherry, S. T., M. H. Ward, M. Kholodov, J. Baker, L. Phan, E. M. Smigielski, and K. Sirotkin. 2001. “dbSNP: The NCBI Database of Genetic Variation.” *Nucleic Acids Research* 29 (1): 308–11.

“The Genetics of Blond Hair.” 2014. Science | AAAS. October 31, 2014. https://www.sciencemag.org/news/2014/06/genetics-blond-hair.

Tretyakov, Konstantin. n.d. *Pyliftover*. Github. Accessed June 3, 2020. https://github.com/konstantint/pyliftover.

Trujillano, D., M. D. Ramos, J. González, C. Tornador, F. Sotillo, G. Escaramis, S. Ossowski, L. Armengol, T. Casals, and X. Estivill. 2013. “Next Generation Diagnostics of Cystic Fibrosis and CFTR-Related Disorders by Targeted Multiplex High-Coverage Resequencing of CFTR.” *Journal of Medical Genetics* 50 (7): 455–62.

“What Effect Do Variants in Coding Regions Have?” 2019. EMBL-EBI Train Online. May 2, 2019. https://www.ebi.ac.uk/training/online/course/human-genetic-variation-i-introduction-2019/what-genetic-variation/what-effect-do-variants.

Xin, Jiwen, Adam Mark, Cyrus Afrasiabi, Ginger Tsueng, Moritz Juchler, Nikhil Gopal, Gregory S. Stupp, et al. 2015. “MyGene.info and MyVariant.info: Gene and Variant Annotation

Query Services.” *bioRxiv*. https://doi.org/10.1101/035667.

———. 2016. “High-Performance Web Services for Querying Gene and Variant Annotation.” *Genome Biology* 17 (1): 91.