FetBASE

Fettuccine Human LINE-1 & HERV Transposable Element Platform

Version 0.1; February 2018

Documentation

What is FetBASE?

FetBASE© – Fettucine Human LINE-1 & HERV Transposable Elements Platform is a fast, reliable and user-friendly interface for exploring Long Interspersed Nuclear Elements 1 (LINE-1) and Human Endogenous Virus (HERV) retrotransposons and their protein expression. This software was developed by the "Fettuccine" student group of Queen Mary University of London for the purposes of a software development group project (School of Biological Sciences and Chemistry – MSc Bioinformatics) under the invaluable guidance of Professor Conrad Bessant (https://bessantlab.org/) and Dr Fabrizio Smeraldi (https://goo.gl/k6jxCr).

The software is based on a complete and updated database of all known LINE1 & HERV repeats of the human genome (Genome Reference Consortium Human Build 38 – GRCh38 genome assembly), as well as all the predicted protein sequences that these genomic loci encode.

One part of the platform's tools is responsible for the interactive exploration of the database. The user can navigate through the tables of LINE1 and HERV families and their translated products, see a visualised distribution of all these elements on the chromosomes and explore their relationships based on their predicted protein sequences. The second part of the platform focuses on the search of user-input queries against the protein records of the database. More specifically, a user can search the entire database with one or multiple protein sequences using a search-box or uploading a file in FASTA format to find out which family member it has been translated from. A direct search of the database with a peptide identification file in mzIdentML or mzTab format is possible, to show and identify if any retrotransposons are being translated in the corresponding file.

The resulting information from the uploaded data is stored in the database to provide information about the retrotransposons for the expression atlas.

1 Software Overview

1.1 Software Architecture

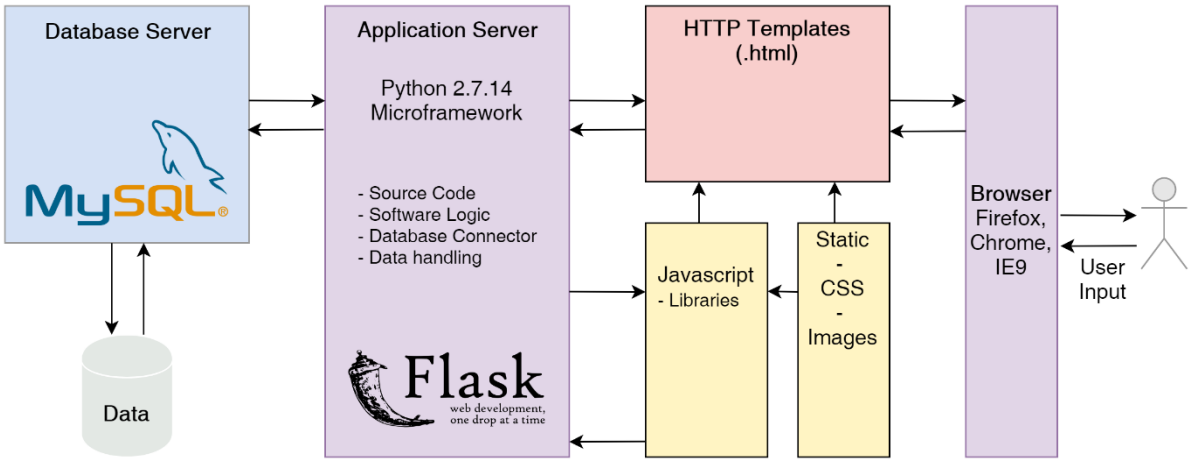

Figure 1. Schematic illustration of the FetBASE software architecture. The platform is based on a MySQL 5.7 database of the retrotransposons and their proteins, which is handled by a MySQL 5.7 database server. The software was developed using Flask Python microframework (Python 2.7.14) which communicates with the database server through mysql Client and runs the source code of the software. Flask communicates and runs along with Javascript to control the behaviour of different objects to the final HTML template. CSS was used to provide a style guide. The software was tested in Google Chrome, Mozilla Firefox and Internet Explorer 9+ and it was completely functional. Image was created using draw.io.

1.2 Software Map

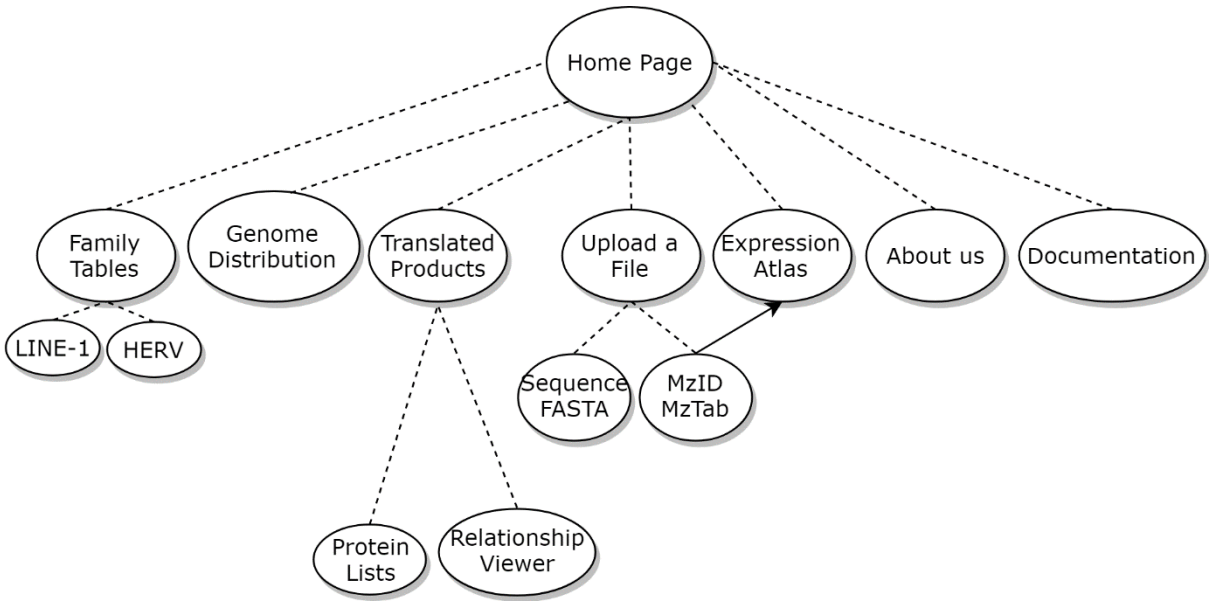

Figure 2. Diagram representing FetBASE navigation map.

1.3 Packages/Systems Used

1.3.1 Flask

Available at http://flask.pocoo.org/ [1].

Flask is a micro web framework written in python which is based on the Werkzeug toolkit and uses the Jinja2 template engine which allows for the use of python-like expressions in HTML documents. This microframework was chosen as the main web development toolkit as it was quick to learn and understand given the time constraints, and the current developers' knowledge in python 2.7 programming language.

Flask was also used as it, by default, protects against the use of cross site scripting (XSS) so malicious scripts that are injected into the website do not compromise the website security, as the software requires and handles input data from the users.

This micro web framework is also compatible with python 3 so if the website needs to migrate from python 2.7, it can occur smoothly.

The following python 2.7 libraries and modules (other than standard) were used to create the website: Flask, MySQLdb , Pandas (https://pandas.pydata.org), NumPy (http://numpy.org), biopython (http://biopython.org), matplotlib (https://matplotlib.org), Pygraphviz (https://goo.gl/JeZwJR).

1.3.2 MySQL 5.7

Available at http://www.mysql.com [2].

MySQL is an open-source SQL management system, distributed by Oracle Corporation, which was used to add, access and process data stored in the FetBASE database. The system is usually known to be fast, reliable, stable and works on all major operating systems. The fact that it is open-source made it easier to modify to suit the needs of the software.

Considering the amount of data that FetBASE database contains (millions of protein entries and thousands of retrotransposonal repeats), MySQL Server was a reasonable choice, given the fast and efficient way of querying even on the fly. It was also very simple to use in conjunction with flask through the use of the powerful MySQL Python Connector and allowed for multiple data tables to be created and read easily. Thus, the MySQL system was key in reaching a robust performance of the FetBASE software, even during intense processes of searching the protein sequences of the database with user-inserted sequences.

MAMP (for windows), AAMP (for linux) was used to run the MySQL server and the web interface (phpmyadmin).

1.3.3 JavaScript libraries

Elements such as having a professional look combined with smooth functionality and interactive features were of high priority during the software development. For this reason, FetBASE includes and runs some useful JavaScript libraries.

JQuery (https://jquery.com):

JQuery is a JavaScript library designed to simplify the client-side scripting of HTML, it was used to add JavaScript functionality into the browser. The features that are available with the use of JQuery are endless. It allows you to manipulate and visualise the webpage in many different ways such as adding animations and processing events after certain specifications are met.

For the distribution of the repeats, JQuery was used to create 24 dropdown buttons which allow the user to select and view the distribution for each chromosome. Using JQuery allows for the information to be displayed on one page instead of multiple pages. JQuery plugins that were used includes:

- DataTables (https://datatables.net/): A Table plugin for JQuery, ideal for projection of large database tables. The main advantage of this plugin is that supports any data

source, server-side processing included, something that really enhanced the speed of loading the protein tables of FetBASE database. Its professional quality, beautiful API and easy customization made it first choice. Furthermore, it provides some useful features such as click-ability, buttons (such as download buttons), search-bar and more.

1.3.4 ChartJS (http://chartjs.org/)

This JavaScript library enables fast engagement of HTML5 based charts with ease. This plugin also provides the use of animated and interactive graphs which constitutes its main advantage combined with easy customization. The loading times of the pages using this plugin were not affected.

1.3.5 HTML/CSS

All websites require HTML to be able to run. Combined with CSS and JavaScript, the possibilities that can occur are limitless. We opted to go for a minimalistic visual approach to make the website seem more professional and refined.
This was done in varying ways such as using CSS and bootstrapping (Bootstrap v4.0.0 https://getbootstrap.com/). Bootstrapping is a popular free frontend framework which allows for efficient creation of websites that are visually pleasing, responsive and have the ability to work with devices with smaller screens such as mobile phones.

The use of bootstrapping has reduced the time required to create the website exactly to our specifications, by allowing faster and seamless addition of content without having to hand create the CSS manually for every object that is displayed in the website.

1.4 FetBASE Database

1.4.1 Getting the Data

1.4.1.1 Getting the HERV and LINE-1 repeats data

USCS Table browser[3] was used to retrieve all known LINE-1 and HERV repeats of the human genome. More specifically, the table of the human repeat annotations, RepeatMasker version open-3.2.7[4] (rmsk table listed in RepeatMasker track/ Repeats group) was used for USCS genome annotation for the Dec. 2013 assembly of the human genome (hg19, GRCh38 Genome Reference Consortium Human Reference 38). 720,871 LINE-1 sequences and 1,001,410 HERV sequences were added to the database with repeat names and HERV classes (named Superfamilies in the Platform) which were defined based on the Repbase classification system[5].

1.4.1.2 Prediction of the protein sequences

Based on the obtained genomic sequences, the corresponding protein sequences were predicted by modifying and using the appropriate python script from Rosalind-Problems GitHub repository (https://goo.gl/wm9CHa) which finds all six Open Reading Frames (ORFs) and translates them. This python code was easy to use and modify as it could identify all ORFs in both directions 5' -> 3' and 3' -> 5' quickly. For LINE-1 ORFs, more than 50 aa where extracted as this is the approximate length of the shortest LINE-1 protein sequence LORF0. For HERVs, only ORFs more than 80 aa in length where extracted as proposed by So Nakagawa and Mahoko Ueda Takahashi[6].

HMMER 3.1b1 (http://hmmer.org) was used to identify which proteins are derived from the retrotransposons. Hmmer is a fast and efficient tool, easy to use and compatible to HPC systems. It can be used to search sequence databases for sequence homologs and for making sequence alignments. The extracted HERV amino acid sequences were searched using the hmmsearch tool with viral motif profiles for gag, pol, pro, env and other accessory proteins separately. Hidden Markov Models

(HMMs) of these viral motif profiles were downloaded from the Pfam (http://pfam.xfam.org/) and Gypsy databases (http://gydb.org). This method led to the identification of sequences that are convincingly derived from HERV retrotransposons with an efficient discrimination between the gag, pol, pro, env and accessory proteins. Similarly, LORF1 and LORF2 protein sequences from UniprotKB [7] (LORF1: Q9UN81and LORF2: O00370) and LORF0 protein sequence consensus [8] were used to run a jackhammer (PSIBLAST-like) search against the extracted LINE-1 protein sequences. This method was selected as it performs a fast and efficient iterative search of a protein sequence vs a protein sequence database. This led to the identification of sequences that are convincingly derived from HERV retrotransposons with an efficient dimination between lorf0, lorf1 and lorf2 proteins.

1.4.2 Sorting the Data

The four main tables of the Database (HERV repeats, LINE-1 repeats, HERV proteins, LINE-1 proteins) have already been pre-determined in Python by combining the initial downloaded data with the predicted proteins and each of their entry has a unique index. All the other tables found in the database were generated using these four tables with the appropriate foreign key constraints being applied. For each HERV retrotransposon, the HERV class (Superfamily) had already been computed by RepeatMasker and the Family was determined through a classification system found in published research work [9], [10]. LINE-1 elements were classified into families based on their repeat names. All the tables that need to be shown in the website are indexed and predetermined in MySQL, something that significantly reduce the loading times of the software.

1.4.3 Database Entity Relationship Diagram

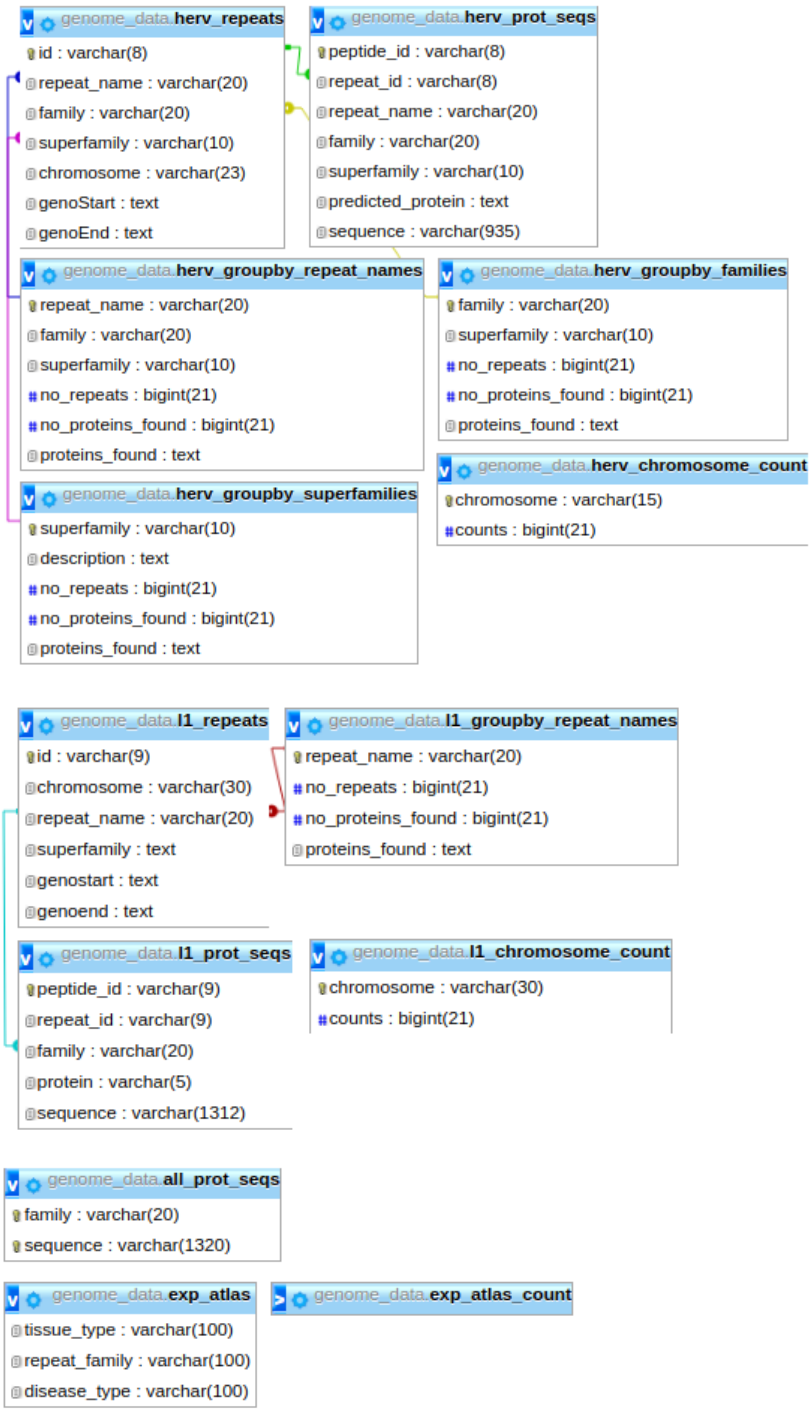

Figure 3. Entity Relationship Diagram of FetBASE Database. Top) HERV retrotransposons database tables all linked to the main table of all listed HERV repeats. Middle) LINE-1 retrotransposons database tables all linked to the mail table of all listed LINE-1 repeats. Bottom) a) "all_prot_seqs" table was constructed by joining the two protein tables (of HERV and LINE-1 retrotransposons) and is searched against user-input sequences b) Tables ready to receive data from Flask regarding the expression atlas. Image was created using phpMyAdmin4.7.

Searching the Database

The table “all_prot_seqs” is searched against user-input sequences. As a result this table was indexed, while the unique combination of the sequence with the family of the retrotransposon was used as primary key. That was a key step, as is facilitates the efficient searching of the table.

MySQL configuration

To efficiently load and search against the protein tables with user-input queries, some of the MySQL configuration settings such as key_buffer_size and max_allowed_packet where increased.

2 FetBASE Features

2.1 Homepage

A JavaScript counter is displayed which queries MySQL to retrieve and calculate the number of HERV and LINE1s in the database and then shows their values to the user.

2.2 Family Table

This feature of the software enables a fast, interactive and efficient visualisation of the tables of the database which provides information on the families of the retrotransposons. HERV retrotransposon grouping was conducted in three levels (Superfamily, Family, Repeat Name), while LINE-1 retrotransposons were only grouped by different Repeat Names. Thus, users can navigate in four main pages, one for each group by selecting “Family table” from the menu and using the buttons on top of the page. The information shown in all tables includes the name of the family, how many times it is found in the genome, the number and name of the proteins found for that family (non-unique). One more column was added to HERV Superfamilies page which provides general information (a “Description” for each

specific superfamily). Two pie charts were also added to this HERV Superfamily page which visualises the genome frequency of each retrotransposon family and the number of proteins that this family encodes. The pie chart visualisation was selected to emphasize to the user, the difference between genome frequency and protein expression among the superfamilies.

All tables were illustrated using “DataTables” JQuery plugin (see 1.3.3 for more information), which is the main reason for the interactivity and the speed of this feature. For this page, the standard configuration for “DataTables” was used. The user can search the tables using the search-bar, sort the columns and navigate through the different table pages using the pagination buttons indicated on the bottom right.

The two pie charts were created using ChartJS (see 1.3.4 for more information). These charts have attractive colours and design, while allowing the user to interact with the graphs, as each pie slice becomes highlighted and shows information on hover.

The data for the tables and the charts are loaded directly from the database on each website’s refresh, contributing to the “updatable” character of the software.

2.3 Distribution table

This part of the application allows users to select a chromosome from a panel with buttons for all chromosomes to view the distribution of HERV and LINE1 repeats. This includes a dropdown function which shows the user the exact number when hovering over each of the 24 ‘Chr’ buttons before selection. This information is displayed through accessing the appropriate tables in the MySQL database and using JQuery to allow for the selection and dropdown functions.

For this visualisation, the R BioConductor libraries karyoploteR (https://goo.gl/CnA3HW) and BSgenome.Hapiens.UCSC.hg38 (https://goo.gl/qAPUJt) were used. The karyoploteR library was implemented to create ideograms for each of the chromosomes 1-22, X and Y. The HERV and LINE1

repeats were mapped to tracks on the ideograms according to their start and end positions on each chromosome. The BSgenome.Hapiens.UCSC.hg38 library was used to set the sequence lengths of the chosen chromosome to avoid incorrect mapping of the repeats. Two tracks were visualised for each Superfamily, one for the mapping of the repeat regions to the appropriate position, customised to allow for overlapping regions due to the number of repeats. The other track plots the density of those repeats based on comparing between overlapping windows of 100,000 base pairs on the chromosome. From this the user can easily see and compare which part of the chromosome contains the most repeats for HERV and LINE1. This process created an ideogram for each chromosome which could be exported as an image from R and integrated into the websites structure.

To improve the ideogram, it was customised to include a track ruler to show the length in base pairs under each chromosome. For this a 'tick' distance of 20 was set which represents the length of 200,000 base pairs. The cytoband names for each chromosome were also added and adjusted to improve and refine the ease of viewing for the user.

The karyoploteR library was chosen as it allows users to easily customize how their data is visualised using a variety of different functions. It can visualise multiple data types and a lot of information in a single image which is important in a project such as this with a large amount of data.

The BSgenome.Hapiens.UCSC.hg38 library was used as the original data from hg38 for the database was obtained through accessing the UCSC genome browser. Hence the right chromosome lengths needed to be used to avoid some repeats not being mapped to some chromosomes.

2.3.1.1 Limitations

During the process of this project, the idea of having an interactive visualisation of the distribution of repeats did not become a reality. This was due to the time restraints and limited resources during the project. Due to the nature of how the distribution of retroelements was implemented, if the website was to be updated with more repeat families, it would have to be updated manually rather than automatically.

2.4 Protein sequences list

Through the page of the protein sequence list, the user can explore the proteins of the retrotransposons by navigating through two big tables (for HERVs and LINE-1s respectively). The tables are accessible via two buttons on the top of the page. Both tables contain a unique id for each protein and information on the family, name and sequence of this protein.

All tables were illustrated using “DataTables” JQuery plugin (see 1.3.3 for more information). Due to the fact that significantly large database tables (~30,000 HERV and ~600.000 LINE-1 proteins) have to be loaded in this page, the server-side processing option of “DataTables” was used. With server-side processing, “DataTables” makes a query to the database server for the data that needs to be shown on the page for a specific time (i.e. paging – Page 1) which is received and converted by Flask into a json object which is projected by “DataTables”. Therefore, all the heavy work is done by the database server, while the software continues to run smoothly. The user can use the search-bar to search these tables for ID, Family or protein name, as the sequence searching is done more efficiently and reliably through the Peptide Sequence Identifier feature of the software. The user can also navigate through the different pages using the pagination buttons on the bottom right.

2.4.1.1 Limitations

However, LINE-1 table is not as fast as it could be due to its large size, something which could be resolved by linking “DataTables” directly to MySQL without Flask’s intervention.

2.5 Relationships Viewer

This part of the website allows the user to click the 2 main buttons (HERV and LINE1) which displays their respective relationship tree image. The trees were created using a couple programs which allowed aligning of the sequences, creating consensus sequences and finally displaying the relationship between the sequences.

Muscle was used to create an alignment followed by Emboss Consensus to create a consensus sequence. Clustal Omega was then used to create a final alignment of the sequences (using EBI online Portal) which also created a newick formatted phylogenetic tree file. This tree file was then processed using an online resource tool called ITOL (Interactive tree of life) to provide the tree Image.

2.5.1.1 Limitations

Originally this part of the application was meant to have interactivity for the user. Showing the peptide-peptide interactions was also an option. However, this did not happen due to unforeseen circumstances during the project.

2.5.1.2 Additional User Functionality – Upload User Tree

The software allows the user to upload a tree file of their own and the software will process the file and output a tree that they can visualise. The only accepted files currently are tree files in the newick format (.nwk, .newick) and .ph files.

Upon uploading a file, a new window displaying the tree file is shown. If the window does not appear, the website automatically displays another version of the tree file in the website itself which is generated after submission of the file.

The Phylo module from Biopython is used to parse the file and then using the matplotlib and phylo module, a tree image is displayed in a new pop up window which can be altered using limited functions such as zooming and panning around.

Before uploading a file, you are given the option to select the DPI (dots per inch) quality of the image to be rendered which is then displayed in the browser. The user is then able to click the image which results in an option to download the image to the user's computer.

If an incorrect filetype is uploaded, the software rejects the file and displays an error message.

One encounter which caused some issues was with a library module called PyGraphViz, necessary for the functionality of this feature, which currently only works with 32bit of python. Unfortunately, using 32bit python would cause memory errors when trying to upload large files (> 300MB) causing the program to crash. This was an issue which currently has not been resolved and is important to discuss as the creation of software is meaningless if it has not been coded for different platforms. Restricting access to a certain type of architecture as experienced first-hand can cause software issues. Upon doing some research, there appears to be unofficial binaries (64bit versions) which do exist and may be able to overcome these issues thanks to Christoph Gohlke (https://www.lfd.uci.edu/~gohlke/) from the University of California. Due to the limited time and the addition of the upload newick tree near the end of the project, we have currently not been able to incorporate PyGraphViz into working with 64bit python.

2.6 Uploading of Files

Allowing users to upload files of their own enables them to search our database and identify family members that the peptide sequence has been translated from.

2.6.1 Peptide Sequence Identifier (FASTA format)

This webpage allows for user input via 2 different methods. The first method allows a user to upload a fasta protein sequence file to the server to analyse the file. The second method allows a user to paste one sequence into a text box before pressing search.

The server first waits for a POST request method to engage before doing any further actions. If a GET request method is used, the normal webpage is served to the client with some data (up to 1000 rows) from the database being displayed. This allows the user to navigate part of the database without any input.

2.6.1.1 Fasta Checker

Due to the possible memory constraints on the host machine, the option was taken to disallow entering in more than one sequence with headers via the text box. Instead the user can upload more than one sequence as a fasta file (including headers) where it can be parsed more efficiently using SeqIO module from biopython.

The text field allows the user to enter any length peptide sequence with spaces and tabs in between and the software can concatenate and parse the data together (for ease of copy / paste use for the client).

If a file is uploaded that contains no data, an error message is relayed back to the user stating the file is empty.

2.6.1.2 Incorrect Filetypes

The program can differentiate between correct and incorrect filetypes. If an incorrect filetype is uploaded, an error will be produced stating the user to upload a correct file with the correct extension.

2.6.1.3 Parsing correct files

If the correct requirements have been met, the peptide data is then queried using MYSQL and if a match has been found in the database, it returns the family name in a table.

2.6.1.4 The programming

The website can differentiate between input via upload box and the search box, conducting different execution of commands depending on the route the information has taken to arrive.

When a post method is received via the text box, the program does a fasta check by looking at the presence of headers and rejects the data if it has been detected. This decision was taken as parsing a file with headers would require more memory and time than would be preferred. The user is instead able to upload the file (with headers) for it to be parsed quicker.

If a post method is received via the upload function, the server moves to the correct file directory to where the file has been uploaded so that it can be read into memory. A few checks are done on the file to ensure that it is an appropriate filetype of the fasta format before being read in. Failure to meet the correct specifications of the file means that the website can provide different errors from incorrect file types to empty file found.

Once the checks have been passed, the Biopython module known as SeqIO is used to parse the file into memory and a further check is conducted to see the length of the sequences. If the length is equal to one, a simple MySQL query is conducted and returns the family name if a match has been found.

The program can also detect whether the total number of sequences is below or above 5000. If it is equal to or below, the program does a simple join of the sequences to create a long query and submits it to the MySQL server. If the number of sequences exceeds 5000, the program divides the sequences into chunks of 5000 and submits the query.

This was done to reduce the wait times and stress on the server. Recent optimisation of the code increased the efficiency by up to 10x.

Jinja2 is a template engine for python that can be coded into HTML pages. For the fasta sequence, jinja2 is used to parse the data that is returned from MySQL. When data is returned, the jinja code executes some code (for and if statements) which then allows for the data to be formatted. Using the Data-tables plugin, this data can then be presented on the webpage in a structured manner where the user is able to use the table search function and go to different pages of the results if it exists.

2.6.2 Mzident / MzTab

This webpage allows the user to upload a file that can be either an mzident or mztab formatted file. The user can also select the tissue type and disease type before submitting the file if the information is known about where the data has been sourced from.

This part of the software can detect the different filetypes that are uploaded and will only accept files with the correct extension.

2.6.2.1 Parsing of files

The type of parsing that occurs is dependent on the type of file extension that is uploaded to the server. The server analyses the extension and differentiates between the accepted and non-accepted forms and then uses regular expressions to extract the information needed. The type of regex used differs between the type of file that has been uploaded. The extracted peptide sequence is then queried in the database via MYSQL and the resulting matches are displayed in a table.

2.6.2.2 Disease / Tissue Types

Before submitting a file, the user can optionally select the tissue and disease type. This is then saved along with the family names that is retrieved from the MySQL query and is used for the expression atlas.

2.6.2.3 Hashing of files

The server conducts a hash check of the contents of the file (the identified peptide sequences) to see if it has been uploaded before. If it has not been uploaded previously, the additional information is saved which contributes to the count tracker that is available on the expression atlas.

To ensure the integrity of the database and the validity of the data (specifically the number of counts / hits of the peptide sequences), files are hashed to create an ID which is then stored in a CSV file. When a file is uploaded, a unique ID is created from the identified peptide sequences using SHA-224 cryptographic hash functions (from the module hashlib). This ID is then searched for in the CSV file and if no match is found, the ID is then appended to the end of the file. A MySQL query is then conducted, saving the family name, tissue type and disease type that was selected by the user. If the unique ID is found in the CSV, the data is not saved into MYSQL DB, however it is still analysed and provides the requested data back to the user.

2.6.2.4 Benchmarking

Benchmarking was conducted to analyse the software's performance when uploading different sizes of MZIDENT, MZTAB or FASTA files (Figure 4).

Table 1. Specifications of the machine used for the benchmarking.

CPU	Intel I5-4570 @ 3.20Ghz processor
RAM	16 GB
Storage	SSD (Project Files) HDD (MySQL Server)
OS	Windows 7 64-bit
Python	Python2.7.14 64-bit

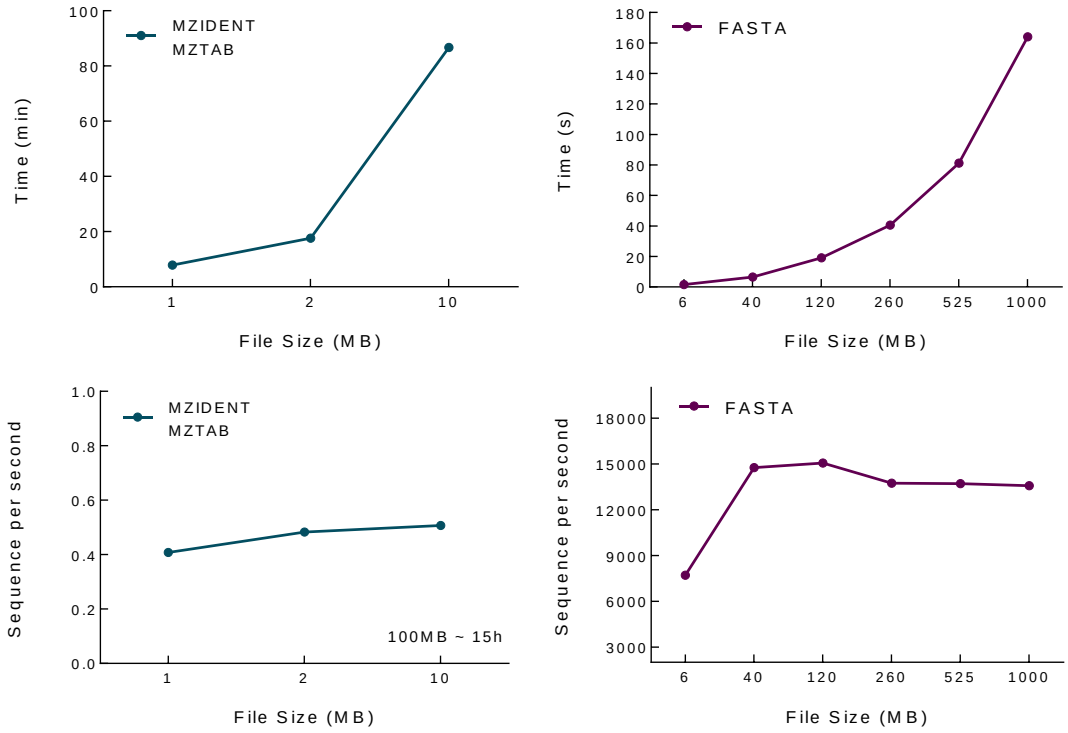

Figure 4. Benchmarking of files from submission to retrieving the results (whole analysis). Top) The total duration of the handling of the file in seconds using files with different sizes. Bottom) Number of sequences analysed per second using files with different sizes. MZIDENT/MZTAB Files (Left) – FASTA Files (Right).

For FASTA files, the results (fig 4) show that the current code created to do the job is adequate for the task required and that the amount of sequences plays a large role in the time taken compared to the size of the file. However, the results for MZID and MZTAB files were not as expected, as the size of the file with sequences needs up to be analysed.

The average times of each of the three parts of the analysis (File upload, Handling Sequences (Python), Resolving MySQL query) were also measured for insight into the allocation of the total loading times (Figure 5). For MZTAB/MZIDENT files the major source of delay was the MySQL queries, while Python reading of the file were very fast. Contrariwise, FASTA files are being read slower by python while MySQL querying times and file uploading seem very fast. These observations indicate that

using regular expressions to find sequences in MZIDENT/MZTAB (instead of using external libraries -such as pyteomics-) had significance in the enhanced performance of this tool. The observed differences in time that MySQL needs to resolve the queries between FASTA and MZID/MZTAB file formats is based on the nature of the query. More specifically, a sequence of a FASTA file needs to be identical with at least one sequence of the database to return a match and therefore, MySQL can use the table's indexing providing a fast search (WHERE sequence = "user-sequence"). Contrariwise, a sequence of an MZID/MZTAB file needs to include at least one sequence from the database to return a match (WHERE INSTR ("user-sequence", sequence) <> 0,) which does not allow MySQL to use the table's indexing in MZID/MZTAB file case.

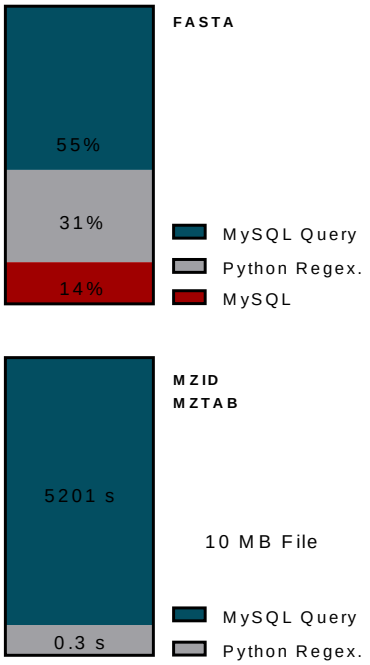

Figure 5. Allocation of total analysis time for MZIDENT/MZTAB and FASTA files.

2.6.2.5 Upload of files (larger files)

Due to the nature of MzId and MzTab files which can typically be up to gigabytes in size, a custom loading screen was added to provide a better aid to the user. The website page is dimmed with the loading bar showing them that the website

is still loading their results. This is required due to some limitations discussed in 2.6.2.6.

2.6.2.6 Limitations

One of the major limitations of this tool is the significantly slow response of the analysis of MzId and MzTab files, due to the reasons discussed in 2.6.2.4. However, the analysis has been checked and proved to be correct and therefore, future software development or anther database choice could make it faster given the fact that this software is a prototype. Another limitation of this tool concerns the hardware used for testing. The type of processors and its clock speed can have a large affect (2 core CPU would run slower than a quad core hyperthreaded CPU). Some programs run on a single core, however a powerful single CPU core would still be able to process the file faster than a slower single CPU core. In an ideal situation, the server would use a Xeon processor along with M.2 SSD drives which would allow for faster read and write access.

2.7 Expression Atlas

This part of the application allows the user to view all the repeats which have been found by the peptide identification in uploaded files from previous users.

This works through interacting with the peptide identification part of the application which inserts information successfully identified from uploaded samples into the MySQL database into a specific expression atlas table. It also inserts and updates the number of each tissue, disease type and total number of samples added into the atlas into separate tables. When the user visits the expression atlas page on the website, the MySQL database is accessed through a query which returns the disease type, tissue type, number of repeats, each individual repeat found with its sequence and relative percentages from the various tables.

This data within the atlas is displayed to the user in two ways for the disease and tissue type; in relation to the percentage of each data type compared to all others

and also by the percentage of each repeat family identified in a tissue/disease type chosen by the user (using a post method). This was achieved through the use two MySQL queries which access two tables within the database to select the relevant information and calculate the required percentages. The data is displayed in interactive tables through the use of the JQuery plugin “DataTables”.

3 Other Features

3.1.1.1 Error 404

The Website can detect when you have entered in an incorrect URL which doesn’t exist and leads the user to a landing 404 error page that allows the user to return to the home page.

4 Technical Solutions / Optimization

The fasta upload function has been optimized vastly with efficiency rate increasing as much as 80%. The first iteration of the code would run a MySQL query for each peptide sequence, this would mean that thousands of queries would be constantly sent to MySQL slowing down the end result.

The code was optimized to be able to send out 1 large query instead (or multiple large queries) so that MySQL only needs to read the table a few times instead of thousands of times. As a result of the code change, a typical 60MB FASTA file with took up to 170 seconds to return a result instead of the old more than 10 minutes depending on MySQL DB state and hardware. Despite the improvements on the speed of the analysis of MzId/MzTab, the nature of the query led to a poor overall performance result as described in 2.6.2.4 & 2.6.2.6.

It is recommended that a user uses 64bit python to run this program otherwise memory errors will be raised due to the nature of how 32bit programs work and the inability to use more memory.

5 General Further Development

This website is a prototype which has the ability to further expand, through implementing more features to create a site which can be used for many purposes related to retrotransposons.

5.1 Link the pages

Regarding the tables used in the software, adding functions such as clickability which redirects the user to other parts of the website such as Protein lists or visualisation could be added. This would help users to further explore the database and provide combined information for a specific entry (i.e. specific protein, specific retrotransposon family). Furthermore, protein tables which handle a large amount of data could be further improved to become even faster with server-side processing. Fortunately, “DataTables” already provides features that could help on the implementation of these ideas and this another reason that this plugin was initially used.

5.2 Updatability

Considering the fact that FetBASE is based on genome data for retrotransposons which can constantly change, it is very important for the Platform to become updatable in a fully automated way. For this reason, the code that was used to handle and store the initial table browser data to the database needs to be combined into a python program which will use the table browser data as input and automatically update the database. Subsequently, triggers have to be added between the database tables in order to update all the related tables when one of the four basic tables are altered.

5.3 Analysis of MzId/MzTab files

As described in sections 2.6.2.4 and 2.6.2.6, the poor performance of the analysis of MzId/MzTab files is one of the most significant limitations of FetBASE. However, this tool is completely functional and returns valid results given the fact that this software is a prototype. Altering the characteristics of the MySQL query used here in order to be able to use table indexes when searching would be a key in enhancing the performance of this tool, as MySQL searching constitutes the slower part of the analysis.

5.4 Ability to Login

One potential extra feature, would be to allow users to register and create accounts. These accounts would then be able to upload their own files to create their own personal expression atlas which would be only available to them unless they allowed public view access. Having a personal account would mean that the user would be able to come and go whenever they like. They would be able to add to their own personal expression atlas and view it from any location in the world as long as they have an internet connected device. Flask has the ability to create pages and return some data dynamically based upon the login details submitted and this would be crucial in being able to gather information for this user from the database.

5.5 Upload Own Tree

A user would have the ability to upload a fasta file where the peptide sequences have already been aligned, and this would then be able to create a tree and provide the user with an image as well as a tree file which they can download.

Further development could also include submitting files that can be automatically aligned (although this would cause server costs to increase) and then a tree file created.

5.6 Interactivity

To make the site more interactive for the user, the implementation of JQuery could be further explored. This would allow aspects such as the visualization of the distribution of retroelements and the relationships between the translated peptides interactive processes for the user. Allowing the user to access even more information in regards to specific repeats and their translated proteins.

5.7 Distribution

To improve the overall visualization and processing of this part of the application, the MySQL database could be directly accessed through R rather than accessing separate csv files which it currently does. Potentially, the creation of the distribution images could be implemented within the application rather than from an outside process and imported. This would allow for automatic updates of the ideograms when updating the database in any way. More information about the individual repeats present on each chromosome could be added below each of the 24 images in an interactive table. Pie charts similar to those on the family table part of the application could also be implemented to show the distribution of repeats for both super families when the user first views the page.

5.8 Expression Atlas

This part of the application could be updated to include even more information about the repeats and uploaded samples to see if there are any distinct relationships between all uploaded data within the atlas. For example, the gender of the patient from which the sample was obtained or more specific disease types. Moreover, the expression atlas tables of the database have to be improved structure-wise to be able to store future possible large amounts of data.